

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА
И ЕЁ ПРИМЕНЕНИЕ

Новое в жизни, науке, технике

Подписная научно-популярная серия

Издается с 1988 г.

Если бы у Марианны была ЭВМ

Операционная система UNIX

1993

1

Новое
в жизни,
науке,
технике

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА
И ЕЁ ПРИМЕНЕНИЕ

Подписная
научно-
популярная
серия

1/1993

Если бы у Марианны была ЭВМ...

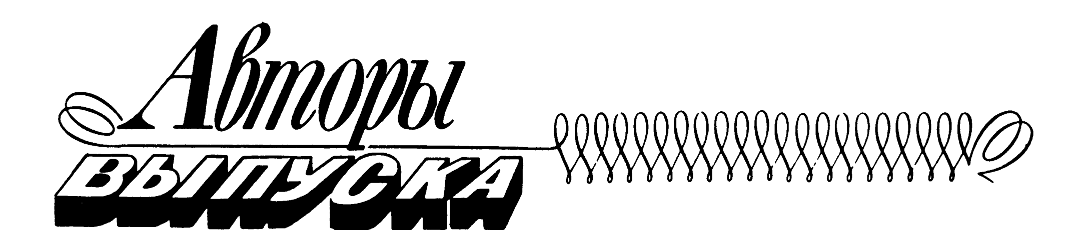
Издается
с 1988 г.

В номере:



Москва
Издательство
«Знание»
1993

ББК 32.81
Е 84



ФРОЛОВ ВЛАДИМИР СЕРГЕЕВИЧ — кандидат технических наук, специалист в области компьютерных систем управления.

ЛЕОНАС ВЛАДАС ВЛАДАСОВИЧ — кандидат физико-математических наук, специалист в области прикладных программ.

БЕЛЯКОВ МИХАИЛ ИОСИФОВИЧ — кандидат технических наук, специалист в области компьютерной техники.

ДЕНИСОВ МИХАИЛ МИХАЙЛОВИЧ — директор по маркетингу СП «ЮНИМАС».

ТОРОПОВ ДМИТРИЙ ИЛЬИЧ — менеджер по маркетингу СП «ЮНИМАС».

КАШИРИН АЛЕКСАНДР ВАСИЛЬЕВИЧ — научный сотрудник Ленинградского института информатики и автоматизации РАН, специалист в области архитектуры ЭВМ.

НИКОЛАЕВ ВЛАДИМИР АЛЕКСЕЕВИЧ — младший научный сотрудник Ленинградского института информатики и автоматизации.

Редактор И.В.КАЩЕНКОВ

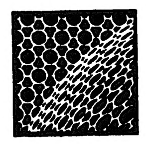
ISBN 5-07-002613-5

К НАШИМ ПОДПИСЧИКАМ

ДОРОГИЕ ДРУЗЬЯ!

Как ни горько,
но после очередного повышения —
вопреки договору —
тарифов «Роспечатью»
все подписные издания
стали убыточными.
Собранных по подписке
средств хватает только на один —
вместо трех — выпуск серии.
Все наши попытки спасти серию,
включая поиск спонсоров
и обращения к высшим
властным инстанциям,
остались безрезультатными.
Благодарим вас за поддержку,
однако она не смогла сохранить
серию.
Приносим свои извинения
и твердо верим:
прощаемся с вами не навсегда.
При первой же возможности мы —
с вашей помощью —
постараемся серию возобновить.

Главная редакция издательства.



А.С.АВАЛОВ

Если бы у Марианны была ЭВМ...

...то она подарила бы ее своему ненаглядному Бето, скажет насмотревшийся телесериала читатель и будет, как всегда, прав. Чему учили Бето частным образом преподаватели (как мы теперь говорим, репетиторы), трудно сказать, по крайней мере, нам этих серий не показали. Если считать, что интеллект Бето определяется наследственными признаками родителей (в интерпретации Михаила Задорнова – «хромосомным набором»), то есть Марианны и Луиса Альберто, которые явно не тяготели к точным наукам, то их сын получал гуманитарное образование. Но в наши дни даже в далекой Мексике и гуманитарию никак не обойтись без компьютера. Представьте, что Марианна и Бето заходят в некий супермаркет под названием «Электронный мир». Чего здесь только нет, глаза разбегаются! Но ту же задачу выбора ПК (персонального компьютера) решает, естественно, с меньшими финансовыми возможностями и российский пользователь.

Итак,

УРОК 1-й. КАК ВЫБРАТЬ КОМПЬЮТЕР «СЕБЕ ПО ПЛЕЧУ»

В комплект любого ПК входят процессор, клавиатура, монитор (дисплей) для отображения текстовой и графической информации, накопители (дисководы) для гибких магнитных дисков (НГМД) и другие элементы, которые будут подробно рассмотрены в уроке 2.

На мировом рынке ПК лидирующее положение занимает фирма IBM. Историческим событием для развития индустрии ПК явился выпуск в 1981 г. ПК под названием IBM PC, получившего широчайшее распространение во всем мире. В 1984 г. начался выпуск ПК IBM PC XT, а в 1986 – IBM PC AT («Advanced Technology» – «продвинутая технология»). Эти три модели широко тиражируются в настоящее время во многих странах (не только в США), а их основные технические характеристики приведены в табл.1.

Таблица 1

Основные характеристики ПК IBM PC

Характеристика	IBM PC	IBM PC XT	IBM PC AT
Разрядность	16	16	32
ОЗУ[1], байт	256 К	640 К	1 М
ПЗУ[2], байт	40 К	40 К	120 К
Дисплей, символы x строки	25x80[3]	25x80[3]	25x80[3]

1) Емкость оперативного запоминающего устройства

2) Емкость постоянного запоминающего устройства

3) Встречаются модификации 25x40

Поколения ПК иногда условно отсчитывают в зависимости от разрядности обрабатываемых ими данных. Так, ПК с 8-разрядными микропроцессорами относят к первому поколению, 16-разрядными – ко второму и 32-разрядными – к третьему поколению.

Крупным шагом вперед фирмы IBM явилась разработка новой серии ПК –

персональных систем типа PS/2. Эти ПК при своем появлении в середине 80-х годов встретили скептическое отношение, но уже через пару лет прочно захватили сначала североамериканский, а затем и международный рынок. В настоящее время фирма производит около 7000 компьютеров PS–2 в день(!). Кроме того, эта модель копируется и тиражируется рядом других зарубежных фирм. Новые модели предоставляют и новые возможности, особенно в области САПР и автоматизации технологических производственных процессов. В то же время они хорошо совместимы с традиционными ПК IBM PC. Роль фирмы IBM как производителя ПК на компьютерном рынке США, а в определенной степени и в мире велика — это порядка 70% от их общего, суммарного, выпуска.

А какие ПК может предложить пользователю отечественная промышленность?

К сожалению, приходится констатировать, что, пожалуй, ни в одной области техники мы не отстаем от стран Запада так сильно, как в области вычислительной техники вообще и производства ПК в частности. От отечественных ученых, конструкторов, технологов, бизнесменов требуются буквально героические усилия для того, чтобы добиться заметного перелома в решении важнейшей на сегодняшний день задачи — насыщения внутреннего рынка современными ПК. Укажем несколько основных семейств отечественных ПК.

Семейство «Электроника». К ПК этого семейства можно отнести настольные компьютеры ДВК–2, ДВК–3, ДВК–3М, ДВК–4, «Электроника–85», «Электроника Т3–29МК», «Электроника БК–0010», «Электроника БК–0010Ш», «Электроника БК–0011» и карманное малогабаритное информационное устройство (МИУ), являющееся электронной записной книжкой-справочником руководителя и бизнесмена. Недостатком всех перечисленных ПК серии «Электроника» является их аппаратная и, самое главное, программная несовместимость с IBM — подобными ПК, которые составляют в настоящее время основной парк отечественных вычислительных устройств.

Семейство «Искра». ПК «Искра–1030М» является настольной ЭВМ класса IBM PC XT и программно совместим с зарубежными и отечественными IBM — подобными машинами. Намеченный к выпуску в начале 90-х годов специализированный ПК «Искра–2030» имеет сходные с ПК «Искра–1030М» характеристики и открывает собой ряд специализированных переносных ЭВМ, выполненных в виде моноблока с открывающейся клавиатурой. Подобное конструктивное исполнение позволяет размещать ПК на ограниченной площади и создает удобства при перевозке или переналадке, что важно при использовании ПК не только в производственных и лабораторных, но и полевых условиях.

Семейство ЕС. Типичным представителем и базовой моделью ПК серии ЕС являются ПК ЕС–1840. Характерный признак всех ПК этой серии — модульность их структуры, которая заключается в том, что все электронное оборудование здесь расчленяется на легко собираемые отдельные модули, связанные между собой системной шиной. Одна из последних моделей ПК ЕС–1842 программно-совместима с IBC PC, и, кроме того, один из разъемов его системной шины позволяет подключать как дополнительные средства платы из состава ПК IBM PC XT.

Семейство учебных (школьных) ПК. В 1982 г. в нашей стране был выпущен ПК «Агат» (аналог американского ПК Apple), процессор которого имел 8 разрядов, быстродействие 300 тыс. опер/с, ОЗУ емкостью 64 Кбайт, ПЗУ — 32 Кбайт. В качестве внешних устройств в нем использовались бытовые телевизор и магнитофон, а также НГМД и термопечатающее устройст-

во. К сожалению, из-за низкой технологической культуры выпускающего предприятия этот ПК себя сильно дискредитировал. Надежность его функционирования была ниже всякой критики.

Относительно малым тиражом выпущен также 8-разрядный ПК ВЭФ «Микро». В качестве основного языка программирования для «Агата» и ВЭФ «Микро» был выбран Бейсик.

В 1987 г. вошел в серию учебный ПК «Электроника УК НЦ» на базе двух 16-разрядных процессоров с быстродействием 800 тыс опер/с. В учебных классах создаются локальные вычислительные сети, как правило, включающие 12 ПК, но в принципе позволяющие объединять до 64 ПК. Языками программирования являются Бейсик, Фортран и Паскаль. Примерно одновременно с «Электроникой УК НЦ» был разработан и комплекс учебной вычислительной техники (КУВТ) под названием «Корвет». Он включает рабочее место преподавателя и 12 рабочих мест учащихся. Для кабинетов информатики и вычислительной техники учебных заведений выпущен ПК «Ириша», а затем чрезвычайно дешевый (в ценах 1986 г.) 8-разрядный ПК «Микроша», представляющий собой одноплатную ЭВМ без каких-либо внешних устройств, но позволяющий подключение любого бытового телевизора и принтера.

В качестве учебных ПК находят также применение бытовые ПК БК–0010 и БК–0011 из уже названного выше семейства «Электроника».

Итак, теперь вы имеете некоторое представление о наиболее популярных зарубежных и практически всех отечественных ПК. Разумеется, ваш выбор будет обусловлен далеко не в последнюю очередь вашими финансовыми возможностями, но все же полезно прислушаться к мнению эксперта: «Желательно, чтобы приобретаемый компьютер был членом семейства компьютеров, совместимых с IBM PC. Покупая такой компьютер, вы получаете доступ к богатейшей библиотеке программного обеспечения, работающего на миллионах аналогичных компьютеров. Обладатель же компьютера с уникальной (в данных условиях) архитектурой вскоре оказывается наедине с уже надоевшей горсточкой программ и с трудом находит «собеседника».

Литература

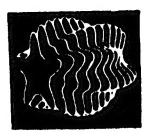
1. Крайзмер Л.П., Кулик Б.А. Персональный компьютер на вашем рабочем месте. – СПб, 1991.
2. Вьюхин В.В. Информатика и программирование в высшей школе. – М., 1992.
3. Боянов К.А. и др. Персональный компьютер. – М., 1989.

ЧИТАТЕЛЮ НА ЗАМЕТКУ

ВИДЕОТЕЛЕФОН С ЭВМ

Японские ученые предложили неординарную концепцию создания нового поколения видеотелефонов, основанную на создании с помощью компьютера искусственной реальности. Описываются принципы проектирования и особенности дисплеев, обеспечивающих «эффект присутствия» абонентов в подобной искусственной среде. Эксперты считают, что новое поколение видеотелефонов решит возникшие ныне проблемы, связанные со снижением стоимости используемой электронной аппаратуры и скоростью передачи видеоинформации.

Журнал «Electron. Mag.» — 1992. — 37, № 1, p. 32—35.



Н.П.РЯЗАНОВ

Что такое «искусственный интеллект»?

...Кризис в Корейской войне достиг своего апогея. «Ястребы» из Пентагона настойчиво требовали от президента пустить в ход ядерное оружие. Обратились за советом к экспертам, вооруженным мощным компьютером. ЭВМ смоделировала возможные последствия атомной бомбардировки и выдала результат: ядерное оружие неприемлемо, ибо ущерб от его применения перечеркивает сиюминутные тактические выводы. Так начинается одна из новых глав в летописях, посвященных проблемам осуществления дерзновенной мечты человека — создания так называемого искусственного разума...

Проблема создания искусственного разума, или интеллекта, захватила ныне не только инженеров, но и гуманитариев. В ней, как в фокусе, сконцентрировались интересы представителей, без преувеличения, всех областей человеческого знания. И не удивительно: именно в этой горячей точке научно-технического прогресса ожидается скорый прорыв, выход на качественно новые рубежи. Активно разрабатывается проблема искусственного интеллекта и в нашей стране.

Прежде всего давайте уточним: понятие «искусственный интеллект» — это не более чем метафора. Так мы говорим для краткости, ибо даже самая совершенная супер-ЭВМ интеллектом не обладает.

Интересно, что понятие «интеллект» определить не так-то просто. По мнению некоторых психологов, интеллект — это то, что оценивается в так называемых интеллектуальных тестах. Многие, наверное, еще в студенческие годы испробовали на себе «тест на интеллектуальность». Испытуемому дается некое подобие мини-анкеты, где предлагается ответить на вроде бы вполне безобидные вопросы («Пьете ли вы по утрам кефир? Любите кошек? Как относитесь к рок-музыке?» и т.п.). За каждый ответ начисляются баллы, которые потом складываются. По сумме, к которой прикладывается соответствующий реестрик человеческих качеств, и определяется ваш интеллект. Случалось, что испытуемый под веселое оживление присутствующих вдруг узнавал, что он «отъявленный скряга, которого больше заботит собственный желудок, нежели состояние здоровья близких».

По-видимому, попытка определить понятие «интеллект» равносильна попытке дать определение, что такое мышление, при ответе на сакраментальный вопрос: «Могут ли машины мыслить?»

Английский ученый А.Тьюринг, видимо, не раз бывал на довольно скучных вечеринках, где гостей, подуставших от своеобразного британского юмора, развлекают «интеллектуальными играми». Кто-то из компании незаметно уходит в соседнюю комнату, а другим гостям предлагают задавать вопросы отсутствующему с тем, чтобы из ответов выяснить, мужчина это или женщина. При этом отсутствующий отвечает не непосредственно, а либо в письменной форме, либо через посредника. Идея Тьюринга (преслову-

тый тест Тьюринга) состоит в том, чтобы посредством подобных переговоров испытать машину на интеллект. Если задающий вопросы не в состоянии установить, общается ли он с человеком или с машиной, и если на самом деле это была машина, то следует считать, что машина обладает интеллектом. Хотя на подобной основе была написана программа для ЭВМ (система ЭЛИЗА по медицинской диагностике) и пациенты, с которыми «беседовал» компьютер, не сомневались: они общаются с врачом, дальше дело не пошло. Сказалось то, что машина Тьюринга моделировала (и то весьма поверхностно) лишь элементы человеческого интеллекта, не была нацелена на получение важных для практики результатов.

В дальнейшем в работах по искусственному интеллекту выделялись два основных направления. Одно из них можно назвать бионическим.

Бионика — научная дисциплина, стремящаяся на основе изучения закономерностей жизнедеятельности биологических организмов помочь конструкторам технических систем. Она как бы расшифровывает патенты живой природы, в которых есть что позаимствовать самому смелому изобретателю. Однако лихая кавалерийская атака на «последний окоп идеализма» — как называли человеческий мозг ура-кибернетики — успехом не увенчалась. Дальше перепева исследований по моделированию нервных сетей дело не пошло. В общем, как говорится, гора родила мышь. Это не значит, конечно, что данное направление следует прикрыть. Но жизнь показала: более эффективным оказалось второе — прагматическое направление. Оно было с самого начала нацелено на практику и не делало попыток моделировать функции мозга. Творческие процессы здесь пытались воспроизвести своим, отличным от человеческого, машинным, т.е. компьютерным, способом. Мы до сих пор точно не знаем, как человек играет в шахматы, но машину запрограммировали, и она прекрасно играет, обыгрывая живых перворазрядников, а в последнее время даже и мастеров. Всемирную известность получили работы советского ученого Р.Х.Зарипова по сочинению на ЭВМ музыкальных произведений. Недавно по центральному телевидению показывалась научно-популярная передача, которая сопровождалась оригинальной мелодией. Вместе с заключительными аккордами на экране поплыли титры: «В передаче прозвучала музыка Андрея Родионова, написанная в содружестве с компьютером». Правда, над работами по искусственному интеллекту навис своего рода антропоморфный туман. В печати, например, каждому из нас не раз попадался набранный крупным шрифтом заголовок: «Машина ставит диагноз». Не машина, а врач ставит диагноз! ЭВМ лишь служит подкреплением памяти специалиста аккумулятором накопленных в данной области знаний, своего рода интеллектуальным усилителем.

Действительно, машина понятия не имеет, что такое диагноз. Она просто гоняет по своим каналам нули и единицы. Даже когда в память (память — тоже метафора, более длинно, но правильнее было бы называть этот блок ЭВМ накопительным устройством) компьютера вводится из внешнего мира информация, то она лишь тогда станет «понятна» для ЭВМ, когда будет закодирована в виде специальных машинных слов. Эти слова представляют самое что ни на есть убожество: наборы единичек и нулей (двоичный код определенной длины). Если нажать кнопку «СТОП» на пульте и заглянуть в чрево машины, то совершенно невозможно понять, чем она занималась: сочиняла ли музыку, решала ли уравнения или резервировала авиабилет.

О подводных камнях, подстерегающих конструкторов интеллектуальных систем, говорит фрагмент следующего диалога между пользователем и

ЭВМ, приведенный на одной научной конференции.

Пользователь: «Сколько стали выплавляет Бенилюкс?» ЭВМ: «Бенилюкс системе не известен». Пользователь в недоумении: ему так много твердили, насколько умна эта машина и вот... Если пользователь немножко упрям и не хочет прекращать диалог, он набирает на терминале расшифровку сокращения: «Бенилюкс — это Бельгия, Голландия и Люксембург вместе взятые». Машина незамедлительно выдает на экране: «Голландия системе не известна». Опять неудача! Все же, не порывая нити разговора, пользователь запрашивает: «Какие западные страны известны системе?» ЭВМ: «Термин «западные» системе не известен». Вновь тупиковая ситуация. Однако пользователь проявляет настойчивость и запрашивает вновь: «Какие капиталистические страны известны системе?» ЭВМ реагирует мгновенно: «Выдаю на экран список: Великобритания, Франция, Бельгия, Нидерланды, Люксембург, Испания...» Наконец-то прояснилась суть затруднений: ЭВМ не знает, что Нидерланды и Голландия — это одно и то же. Теперь пользователь на верном пути и по-иному дает расшифровку сокращения: «Бенилюкс — это Бельгия, Нидерланды и Люксембург вместе взятые». ЭВМ: «Информация понятна» — и мгновенно высвечивает на экране долгожданный ответ.

Однако далеко не всякий пользователь проявит подобную настойчивость, или, может быть, точнее покладистость. Вероятнее всего он перестанет верить системе и, что еще хуже, в эффективность подобных компьютерных систем. Именно поэтому еще на этапе проектирования систем с искусственным интеллектом следует обязательно учитывать профессиональные интересы пользователя.

В современных условиях, когда человеку становятся подвластны могучие силы природы (он сам становится, по выражению академика В.И.Вернадского, настоящей «геологической силой»), когда от его действий и, если хотите, интеллекта зависит здоровье целых поколений, да и сама жизнь на планете, неизмеримо возрастает ответственность тех, кому доверено сказать «Да» или «Нет», т.е. принять окончательное РЕШЕНИЕ. Появилась даже аббревиатура ЛПР — лицо, принимающее решение.

Ученые до сих пор бьются над составлением математической модели святая святых — процесса принятия решения. Их голубая мечта — «подключить» к человеческому мозгу в буквально решающие мгновения интеллектуальный усилитель — быстродействующий компьютер. Мыслительная деятельность ЛПР относится к явлениям, трудно поддающимся формализации, здесь многое зависит от опыта. Чтобы еще более повысить точность принимаемого решения, действуют по принципу «одна голова — хорошо, а две — еще лучше». Вспомним знаменитый совет русских военачальников в Филях накануне Бородинской битвы. Выслушав мнения генералов, Кутузов принял окончательное решение: «Быть посему!» Подобное объединение нескольких ЛПР давно используют в медицине (консилиум врачей) и даже спорте (судейская бригада в фигурном катании). По существу, здесь человек применяет проверенный веками способ объединения формальных и неформальных процедур в одной процедуре, называемой экспертизой. Экспертные оценки позволяют переплавить, слить в единый коллективный интеллект общечеловеческий опыт, социальную память и профессиональное мастерство экспертов. Действительно, каждый из девяти судей, оценивающих выступление на льду, скажем, танцевальной пары не имеет ни спидометра, оценивающего скорость движения фигуриста, ни штанген-циркуля для проверки геометрической правильности узоров. Он оперирует с качественными характеристиками: «красивее», «сложнее», «музыкальнее». И вот за-

тем на световом табло вспыхивает итог интеллектуальных изысканий спортивного судьи — результирующая, точнее, две результирующие оценки: «за техническое мастерство» (т.е. за логику исполнения) и «артистичность» (т.е. за вызванные им эмоции). Существует довольно сложный многоступенчатый алгоритм подсчета зачетных баллов на компьютере по различным программам, утвержденный конгрессом ИСУ (Международным союзом конькобежцев).

Другой, более сложный, пример. Чем ЛПР должен руководствоваться при выборе стратегических направлений научно-технического прогресса, составлений экономических и социальных программ, нацеленных на перспективу? Действительно, как распределить денежные средства, материальные и людские ресурсы по разрабатываемым научным направлениям? Очевидно, вариант «Всем сестрам по серьгам» не подходит, требуется выделять наиболее важные, приоритетные направления. Но неприоритетные на сегодняшний день «науки — Золушки» могут завтра выйти на передовые рубежи научно-технического прогресса, и мы окажемся в проигрыше. Не обеспечив ускоренного развития таких «Золушек», мы будем вынуждены закупать оборудование за границей, тратить немалую валюту, нести и другие потери. Как видим, задача отнюдь не проста. И вот здесь на помощь должен прийти искусственный интеллект, материализованный в виде так называемых экспертных систем.

«В наши дни начался настоящий бум экспертных систем, — говорит доктор технических наук профессор Д.А.Поспелов. — Во всем мире создано уже около сотни разнообразных экспертных систем, используемых в различных областях медицины, в экспериментальной химии, фармакологии, геологии, археологии. Экспертные системы начинают внедряться при автоматизации проектирования и в экономике, в системах автоматизации научных исследований и в истории — везде, где специалистам приходится иметь дело с большими объемами знаний, носящими, как правило, неформальный характер».

В свое время Всемирная организация здравоохранения распространила копии историй болезни (с многочисленными анализами, кардиограммами и др. объективными данными) нескольких пациентов из Скандинавии. К каждой копии была приложена просьба к ведущим специалистам дать заочный анализ заболевания. В СССР копии были размножены и разосланы крупным ученым-медикам. Потом они встретились в Москве для обсуждения своих диагнозов. Была составлена и выставлена на всеобщее обозрение сводная таблица суждений врачей-экспертов. Поразительным оказалось то, что для каждого пациента имелся внушительный разброс суждений о его самочувствии (от «практически здоров» до «заболевание икс в самой тяжелой форме»). Авторы диагнозов выходили на трибуну и обосновывали свою точку зрения. Вопреки житейскому здравому смыслу эксперты, используя по сути одни и те же исходные данные, приходили не просто к разным, но зачастую к противоположным выводам. Стало очевидным, что даже большого числа формализованных, объективных, пусть и весьма точных данных для постановки диагноза далеко не достаточно. Такую ситуацию, в общем-то, уже предвидел известный французский летчик, писатель и, как мы теперь знаем, изобретатель Антуан де Сент-Экзюпери: «Я верю, настанет день, когда больной неизвестно чем человек отдастся в руки физиков. Не спрашивая его ни о чем, эти физики возьмут у него кровь, выведут какие-то постоянные, перемножат их одна на другую. Затем, сверившись с таблицей логарифмов, они вылечат его одной-единственной пилюлей...» (В то время еще не было компьютеров и основным инструментом прикладной математики являлась логарифмическая

линейка.) «И все же, — продолжает Экзюпери, — если я заболею, то обращусь к какому-нибудь старому сельскому врачу. Он взглянет на меня уголком глаз, пощупает пульс и живот, послушает. Затем кашлянет, раскурив трубку, потрет подбородок и улыбнется мне, чтобы лучше утолить боль. Разумеется, я восхищаюсь наукой, но я преклоняюсь и перед мудростью...»

Мудрость в наши дни у специалистов по искусственному интеллекту называется банком или базой знаний. Знания и данные — не одно и то же. Данные — это первичная, пассивная информация, вводимая в ЭВМ от датчиков робота-манипулятора или по пикам кардиограммы пациента. Знания же активны, противоречивость в них заставляет человека стремиться к ее преодолению. При построении интеллектуальных систем воспроизводится это важнейшее свойство базы знаний человека. Здесь исследователю приходится вторгаться не только в медицину, но и в теорию языкознания и семантических сетей с их основополагающими структурами типа «сценарий», «слот» и «фрейм». Знания в отличие от данных нельзя просто перекачать как информацию на магнитные диски ЭВМ из справочников, научных трактатов и прочих умных книг. Их может сообщить машине только человек, причем специалист высокой квалификации, эксперт. В результате в компьютере создается семантическая структура вроде кристаллической решетки, только на стыках решетки остаются незанятые места. Эти места и заполняются соответствующими вполне определенными содержательными понятиями. Возникает система, обладающая качественно новыми свойствами. Она реагирует на изменение ситуации вполне определенным образом. Если же заменить некоторые ячейки другими понятиями, то оценка ситуации и соответственно поведение системы существенно изменится.

Учитывая актуальность создания подобных систем самого различного предназначения, в свое время Президиум АН СССР образовал специальный Научный совет по проблеме «Искусственный интеллект» во главе с академиком Г.С.Поспеловым. Этот совет активно сотрудничает с академическими структурными подразделениями, министерствами, ответственными за выпуск компьютерной техники. Но по мнению ученых, этого мало. Надо смелее подключать и нашу молодежь. Уже образовалось ядро энтузиастов в городе Переяславле. Но и этого недостаточно! Необходимо в каждом НИИ, в каждом вузе и даже в школе создавать группы энтузиастов по искусственному интеллекту. Важно поддержать их новаторский, интеллектуальный поиск.

ЧИТАТЕЛЮ НА ЗАМЕТКУ

БИОГРАФИЯ ПЕРВОГО «АРХИТЕКТОРА» ЭВМ

В ГПНТБ хранится оригинальный справочный материал, посвященный жизни и деятельности известного американского ученого Джона фон Неймана (1903—1957), выходца из Венгрии, доктора математики, профессора Принстонского университета. Рассказывается об его участии в знаменитом Манхеттенском проекте по созданию первой атомной бомбы, о проектировании и применении ЭВМ, о его научных работах, связанных с экономико-математическим описанием процессов товарообмена в вариантах монополии, олигополии и свободной конкуренции.

РЖ ВИНИТИ
«Техническая кибернетика». — 1992. — № 10. — С. 1.



Любой ученый или поэт знает, как сладок и вместе с тем так редок волнующий миг озарения. Как долго приходится ждать, пока «в окно вдруг постучится Синяя птица удачи». А нельзя ли ускорить прилет этого синекрылого ангела! «Можно и нужно», — утверждает советский ученый-изобретатель А.Зенкин.

В.С.ФРОЛОВ

Озарение при помощи... ЭВМ

Мы как-то привыкли к тому, что законодателями мод по части компьютеров являются для нас кристианы диоры, проживающие в США или Японии. И мы добросовестно учимся у них, перенимаем монтаж интегральных схем, обучаем студентов программировать сверху вниз, заимствуем новую архитектуру мини-ЭВМ. В общем, мы догоняем. Но идущий по следу не обгонит! А может быть, нам найти собственный путь и здесь? Ведь в ближайшие годы, считает академик А.Самарский, трудно ожидать, что наш рынок насытится компьютерами новейших поколений, вдобавок доступными среднестатистическому трудящемуся. Механическое копирование американского и японского опыта далеко нас не выведет. Наш козырь — интеллект! Да, да — интеллект, который не сломили жестокие годы культа и волюнтаризма, который не зачах от парникового эффекта расслабляющих лет застоя. Естественно, интеллект выступает ныне в качественно ином мире, в другом «климате», созданном НТР. Компьютеры сошли с полотен художников-фантастов и обрели реальную жизнь в научных лабораториях, заводских цехах, школах и даже на рабочем столе писателей и композиторов. В печати, по радио и телевидению зазвучали как позывные нового мира два магических слова: искусственный интеллект.

Проблема создания искусственного интеллекта волнует не только инженеров. В ней, как в фокусе, сконцентрировались интересы представителей, без преувеличения, всех отраслей знания — естественных и гуманитарных. И неудивительно: именно в этой горячей точке ожидается скорый прорыв — выход на качественно новые рубежи. Активно разрабатывается проблема искусственного интеллекта и в нашей стране.

«Достаточно указать на вклад ученых Московского университета», — говорит профессор Д.А.Поспелов.

Недавно вышла из печати его монография «Ситуационное управление» и немедленно разошлась — так много созвучных нашему времени оригинальных идей содержится в монографии. По мнению Дмитрия Александровича, работы сотрудника университета А.А.Зенкина в области искусственного интеллекта носят пионерный характер.

«О них надо писать и писать, ибо в мире подобных исследований практически нет, — продолжает профессор Поспелов. — Новые результаты из теории чисел, полученные таким способом А.А.Зенкиным, по-моему, впечатляют...»

Выдержка из рекламного проспекта Выставки достижений народного хозяйства СССР: «Главная особенность и главное достоинство разработанной доцентом МГУ Зенкиным диалоговой комптьютерной системы ДСТЧ состоит в том, что она позволяет визуализировать на экране графического дисплея в максимально удобной форме информацию о структуре абстрактных теоретико-числовых объектов... Расшифровка двумерных изображений этих объектов позволяет формулировать гипотезы и теоремы об их математи-

ческих свойствах, подобно тому как расшифровка инфракрасных, ультрафиолетовых и т.д. спектров позволяет химику формулировать химические теоремы о пространственной структуре соответствующих молекулярных систем... Визуализация (и озвучивание) структуры абстрактных математических объектов существенно упрощает и делает наглядной интерпретацию известных результатов классической теории чисел. Более того, непосредственно по изображению на экране дисплея (а в ряде случаев и на слух) можно увидеть (и услышать) новые математические факты и сформулировать принципиально новые математические утверждения... Система ДСТЧ может также использоваться для развития творческого воображения и научной интуиции студентов...»

МГУ сегодня — это буквально город в городе, больше сотни зданий и сооружений, раскиданных по разным микрорайонам.

«Как нас найти?.. — переспросили меня на другом конце телефонного провода. — Знаете главное здание МГУ? А где химический факультет? Так вот, соедините прямой его вход с памятником Ломоносову. Мы как раз на этой прямой за университетской чугунной оградой...»

Еще на подходе к лаборатории ухо улавливало обрывки каких-то мелодий, будто кто-то играл на арфе или ксилофоне. То пел компьютер, точнее синтезатор, подключенный к персональной ЭВМ. Ее хозяин — доцент Александр Александрович Зенкин, воспитанник химического факультета. Еще будучи студентом, увлекся кибернетикой, освоил немало типов ЭВМ, создал первый в стране компьютерный фильм. Ныне плодотворно работает в области искусственного интеллекта.

«Хотите послушать «симфонию Гаусса»?» — предложил Александр Александрович.

Вопрос несколько озадачил. Насколько мне было известно, Гаусс, хоть и был в свое время признан королем математиков, никаких музыкальных произведений не писал.

«Эту симфонию «исполняют» его теоремы», — пояснил Александр Александрович.

Зенкин включил компьютер и, словно заправский пианист (позже я узнал, что он учился в музыкальной школе), стал «играть гаммы» на пластмассовой клавиатуре. Засветился экран и, как на футбольном электрифицированном табло, вспыхнули ровные ряды строчек.

«Так называемое меню. На сегодня выбираем Гаусса, хотя, смотрите, есть еще теоремы Эйлера, Лобачевского, Лагранжа, Варинга и других математиков.

Экран дрогнул и в мгновение ока покрылся стремительно бегущими квадратиками — красными и светло-зелеными. Компьютер «запел» — лаборатория наполнилась электронной музыкой, создаваемой синтезатором. Правда, еще трудно было уловить какую-то определенную мелодию, ведущую музыкальную нить. Во время хоккейных баталий такой музыкой обычно потчуют зрителей, когда шайба улетает за борт.

«Это не удивительно, — Зенкин кивнул на экран дисплея, — там пока натуральный ряд чисел: один, два, три и так далее. Так что и мелодия не блещет оригинальностью. Теперь, смотрите, вводим в компьютер теорему Гаусса...»

Разноцветные квадратики умерили свой бег, постепенно собираясь в вертикальные столбики. Наконец на дисплее возник четко ранжированный геометрический рисунок — своего рода визуальный образ теоремы. Одновременно изменился характер мелодии — она словно стала прозрачной, послышалась арфа, ксилофон. Пела электроника. Конечно, то был не Чайковский и даже не Челентано, но, поверьте, трудно было отделаться от ощущения какой-то фантастичности всего происходящего. С нами говорила теорема Гаусса!

— Но ведь, — спрашиваю я, — мелодию исполняет бездушный синтезатор, который и понятия-то не имеет ни о каком Гауссе?

— Да, компьютер, имеющий синтезатор звука, легко научить исполнять любую мелодию, — соглашается Александр Александрович. — Я же написал специальную программу, которая исполняет только классику — произведения великих математиков. Громкость, тембр и высоту тона синтезатор позволяет выбирать по вкусу. А вот ритм, расстояние между различными нотами однозначно определяются структурой исполняемого математического произведения, то есть константами Природы. Поэтому любая теорема имеет свою визуальную и музыкальную визитную карточку. Музыку, скажем, теоремы Эйлера не спутаешь с музыкой теоремы того же Гаусса или Лагранжа.

Зенкин выбрал для начала раздел высшей математики, известный как теория чисел. Казалось, в ней все давно изучено вдоль и поперек. Однако интерес к теории чисел резко возрос с началом эры космических полетов, быстродействующих ЭВМ и промышленной робототехники. И тут выяснилось, что в этом разделе науки немало еще не решенных задач и даже целых проблем. Скажем, проблема Ферма существует уже более двухсот лет, но, несмотря на усилия величайших математиков, до сих пор не найдено ее полное решение. Или так называемая проблема Варинга. В свое время этот ученый высказал гипотезу: любое натуральное число можно представить в виде суммы таких же натуральных чисел, возведенных в нужную степень. Причем какой бы ни была степень, все равно слагаемых нужно не бесконечно много, а вполне определенное (конечное) количество. Доказать или опровергнуть эту гипотезу никому не удавалось вплоть до начала XX века.

«Но проблема потому и называется проблемой, — говорит Зенкин, — что ее решение вовсе не означает закрытие проблемы и потерю к ней интереса. Так, крупный советский математик академик И.М.Виноградов, исследуя проблему Варинга, создал свой знаменитый метод тригонометрических сумм. С его помощью решено множество других задач теории чисел. К настоящему времени открыть что-либо новенькое в проблеме Варинга практически невозможно. Так скажет вам любой математик. Но не ... компьютер. Оказывается, компьютер имеет на этот счет свое собственное мнение».

Зенкин вновь прошелся по клавиатуре терминала персонального компьютера. На экране дисплея возникло нечто вроде бесконечной целлулоидной ленты, размеченной разноцветными квадратиками. Лента не вмещалась во весь экран, и компьютер стал «нарезать» ее на куски, аккуратно «подклеивая» их друг под другом. Казалось, весь экран будет забит хаосом разноцветных квадратиков. Однако Зенкин изменил масштаб, и теперь человеческий глаз схватил зрительную суть закономерности — парабола! Да, цветные квадратики как бы материализовали новую, доселе не известную закономерность в теории чисел — положенную на бок параболу. Математическое свойство параболичности, оказывается, вошло в плоть и кровь рассматриваемых числовых последовательностей, является инвариантным, т.е. не меняется ни при каких обстоятельствах. Так обнаружился совершенно новый класс чисел в тысячекратно изученном математиками натуральном ряду. Соавтором этого открытия может быть назван компьютер.

Анализ свойств открытых множеств, выполненный опять-таки с помощью компьютера, дал возможность установить их математическую структуру, что позволило впервые сформулировать и доказать обобщение знаменитой теоремы теперь уже для любых условий. Результаты исследования были опубликованы в докладах Академии

наук, а созданная ученым диалоговая компьютерная система с успехом демонстрировалась на Выставке достижений народного хозяйства.

«Собственно, в том, чтобы визуализировать научные абстракции и тем самым сделать их доступными для нашего живого созерцания, — признается Зенкин, — нет ничего нового. Действительно же новое заключается в том, что один из уникальнейших элементов современной информационной технологии, а именно так называемая интерактивная компьютерная графика, позволяет делать это на качественно новом уровне, резко повышает эффективность научного труда. Графика может не только убедительно проиллюстрировать суть, смысл сложного теоретического вопроса: она позволяет — и не так уж редко — увидеть новые, неожиданные ракурсы, казалось бы, хорошо известного, высказать новое соображение, мысль, идею».

Действительно, давно подмечено, что график, рисунок не просто служат иллюстрацией какого-то явления или теории. Нет, они подчас высвечивают невидимые доселе грани проблемы и даже подсказывают верный путь к ее решению. Причем наилучшее, или, как говорят математики, оптимальное, решение подчас принимается эвристически — на интуитивном уровне. Иными словами, процесс принятия решения (а в науке это может быть выдвижение новых гипотез) происходит творчески, ему тесно в прокрустовом ложе непререкаемых истин, жестких логических конструкций и пресловутого здравого смысла.

Зенкин разработал методику интенсификации святая святых — глубинных творческих процессов, повышения к.п.д. интеллекта. Ученый рассуждал так: основа любого творчества — неуловимая интуиция, озарение, «инсайт», как говорят психологи. Можно ли ускорить появление инсайта? Видимо, для этого необходимо как-то воздействовать на человеческий мозг. Прямой канал, соединяющий мозг с внешним миром, — наш глаз. Глаз — не просто миниатюрный оптический прибор, подаренная нам природой совершенная «фотокамера» и т.д. Медики прямо говорят: глаз — это вытолкнутый наружу мозг. Именно поэтому «лучше один раз увидеть, чем сто раз услышать» — девяносто процентов информации о внешнем мире поступает из наших глаз, так называемого зрительного анализатора. В результате в полушариях головного мозга создается не просто геометрический аналог наблюдаемого объекта, а его чувственный образ. До сравнительно недавнего времени считалось, что два полушария головного мозга природа придумала для надежности и взаимозаменяемости, как, скажем, пару рук или пару легких. Внешне одинаковые, как близнецы, левое и правое полушария неожиданно оказались совсем разными по своим функциям. С точки зрения кибернетики это две различные системы восприятия сигналов внешнего мира, их обработки и планирования поведения. Упрощенно можно сказать, что левое полушарие позволяет человеку осуществлять рационально-логические процедуры. Все они могут быть словесно описаны и, следовательно, алгоритмизируемы. Совсем иное дело — правое полушарие. Это мир чувств, эмоций, воображения, интуиции и образного мышления.

— Такая дуальность, — считает профессор Д.А.Поспелов, — основа нашего мышления. О функциях левого полушария, ответственного за рационально-логические функции, известно сравнительно немало. Но как проникнуть в тайны правого полушария? Оно, к сожалению, «молчаливо», и о протекающих там процессах можно только догадываться. Это и объясняет «левополушарный» крен в теории интеллектуальных систем и роботов. Пока мы делаем и вкладываем в них то, что нам более понятно, объяснимо и алгоритмизуемо...»

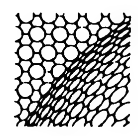
По мнению Зенкина, «левополушарный» крен характерен не только для

систем с искусственным интеллектом, но для всего научного познания, начиная с того «исторического момента», когда наука официально отделилась от искусства, и вплоть до наших дней. Это значит, что, несмотря на свой фантастический прогресс, современная наука использует пока лишь пятьдесят процентов мощностей человеческого мозга!

«И вот интерактивная компьютерная графика, — развивает Зенкин свою идею, — позволяет эффективно включить в работу и вторую (причем наиболее творческую!) половину мозга — правополушарную. Таким образом, интерактивная компьютерная графика — это не просто средство иллюстрации и визуализации, увеличивающее наглядность представления уже известной информации, а принципиально новый, мощный и эффективный инструмент научного познания, инструмент, я бы сказал, расширенного воспроизводства качественно нового научного знания. Он предоставляет исследователю уникальную возможность непосредственного общения уже не столько с ЭВМ, как мы уже к этому привыкли, а ... с самой исследуемой научной проблемой. Суть ее — сколь бы абстрактной она ни являлась по своей природе — в максимально наглядной цветомузыкальной форме отображается на экране дисплея».

Крупнейшие специалисты в области искусственного интеллекта отмечают, что Зенкину уже удалось материализовать свои идеи и создать специальные графические средства, осуществляющие подсказку при поиске новых закономерностей в наблюдаемом материале. Пока в качестве такого наблюдаемого материала выступает натуральный ряд чисел с заранее заданными свойствами, интересующими исследователя, выделяются они с помощью цвета на экране дисплея в виде некоторого закономерного рисунка, задаваемого выделенными цветом числами. С помощью интерактивной когнитивной графики (ИКГ) автоматизируется характерный для многих видов научной деятельности этап — поиск возможных закономерностей, которые могут стать основой принципиально новых теоретических утверждений.

Таким образом, значение ИКГ далеко выходит за рамки чистой математики, или так называемых точных наук. Детище Зенкина сулит произвести настоящую революцию в системе образования. Действительно, ИКГ не только позволяет визуализировать суть, смысл, само содержание сложных абстрактных научных понятий и теорий. Более того, характерную для сегодняшних форм обучения статику изложения учебного материала она заменяет живой динамикой идей. Представьте себе: вместо нудного словесного и символического описания законов и теорий — демонстрация реальной жизни этих законов! Все это, по мнению Зенкина, сделает понятным и доступным даже школьникам то, что сегодня преподается как высшая премудрость («только для посвященных!») в вузах и институтах повышения квалификации. Социальный же эффект таков: за счет сокращения сроков овладения этой премудростью увеличивается продолжительность активной интеллектуальной жизни всего населения сразу лет на 10 (сейчас увеличение продолжительности жизни лишь на год требует десятилетий).



Основанная на концепции монопольной однопрограммности, ОС MS-DOS в настоящее время вызывает у пользователей большее неудовольствие, чем, например, недостаточно высокая вычислительная мощность компьютера.

В.В.ЛЕОНАС

Из истории создания ОС UNIX

Одной из очень часто встречающихся задач, решение которой крайне затруднено при использовании ПК, работающих в монопольном однопрограммном режиме, является задача разделения информации. Такая задача возникает, например, при необходимости автоматизации учрежденческой деятельности, для чего, как правило, оказывается необходимым использование компьютерных сетей и централизованных баз данных. Наиболее удобным средством для организации централизованных баз данных в компьютерных сетях является файл-сервер. Он представляет собой узел компьютерной сети, в функции которого входят хранение больших объемов информации и обеспечение доступа к ней из остальных узлов этой сети. Компьютеры, используемые для создания на их основе файл-серверов, должны обладать достаточными информационными и вычислительными ресурсами. Наиболее просто и эффективно функции файл-сервера реализуются при использовании компьютера, работающего под управлением многопользовательской многопрограммной ОС разделения времени. В качестве такой ОС чаще всего используют ОС ONIX или подобные ей операционные системы.

Даже фирма IBM, в течение длительного времени не признававшая идеологию ОС UNIX, предлагает теперь подобные ей ОС: ОС PCIX — пользователям своих ПК типа XT и AT и ОС AIX — пользователям ПК модели 80 из семейства PS/2. Новое семейство RISC компьютеров IBM, получившее название RS 6000, полностью ориентировано на использование ОС UNIX.

В настоящее время сочетание слов «персональный компьютер» и «операционная система (ОС) UNIX» воспринимается уже совсем не так фантастически, как лет 5—10 назад. Действительно, в те годы сообщение о разработке версий ОС UNIX для 8-разрядных персональных компьютеров (ПК) на основе микропроцессоров Intel 8080 и Z8000, а также для младших моделей 16-разрядных ПК серии LSI—11 могло вызвать в лучшем случае снисходительную усмешку. Появление версий ОС UNIX для ПК на основе микропроцессоров Intel 8086 и Intel 8088 (например, ОС XENIX) продемонстрировало лишь принципиальную возможность того, что ОС этого семейства могут работать (неважно, плохо или хорошо) на 16-разрядных ПК. Опыт разработки ПК на основе микропроцессора Intel 8088 и версий ОС UNIX для них (в первую очередь речь идет, конечно, об ОС XENIX) показал реальность использования ОС этого семейства на 16-разрядных ПК. Однако только появление компьютеров на основе микропроцессора Intel 80386, по-прежнему называемых персональными, но которые язык уже (или, может быть,

еще?) не поворачивается так называть, сделало, по-видимому, сочетание слов «ПК» и «ОС UNIX» неразрывным на будущее. Этому также способствовало успешное завершение процесса стандартизации ОС UNIX (практически разработан процесс стандартов POSIX).

Возникает законный вопрос: чем обусловлена необходимость создания новых ОС для ПК, если есть такая популярная ОС, как MS-DOS, для которой разработано фантастически большое количество пакетов прикладных программ из самых разных предметных областей? Не вдаваясь в обсуждение ситуации с «виртуальной» (она как бы есть, но ее как бы и нет) операционной системой OS—2, попытаемся ответить на поставленный вопрос.

Даже простое сравнение ОС UNIX и MS-DOS показывает, что первая из них по своим возможностям намного богаче второй. Данное обстоятельство определяется в первую очередь различной природой этих двух операционных систем; как уже говорилось, ОС UNIX является многопользовательской многопрограммной операционной системой, а MS-DOS представляет собой однопользовательскую однопрограммную операционную систему (строго говоря, ее и операционной системой назвать трудно — это, скорее, монитор). Поклонники MS-DOS, не знакомые с настоящими многопользовательскими многопрограммными ОС, обычно возражают против этого, ссылаясь в качестве примера на среду MS Windows, которая не является по-настоящему многопрограммной средой (о многопользовательской среде здесь речи вообще идти не может). Кроме того, в OS UNIX максимальная длина программ практически не ограничена — она может достигать 16 Мбайт, в то время как в MS-DOS эта величина не превышает 640 Кбайт. Наконец, такие поддерживаемые OS UNIX функциональные возможности, как электронная почта, сетевые средства, управление планированием выполнения программ, средства автоматического вызова программ на выполнение через заданные интервалы времени или в определенные моменты времени, контроль доступа к информации, различные многооконные системы (например, системы Multiview и Xsight, соответствующие стандарту X11) и другие, делают эту ОС привлекательной для пользователей.

Согласно формальной классификации ОС UNIX следует отнести к категории многопользовательских многопрограммных операционных систем, работающих в режиме разделения времени. Однако гибкость ОС UNIX обусловила удобство ее адаптации для работы как в режиме реального времени (примером такой адаптации может служить ОС UNOS, разработанная фирмой Charles River Data Systems), так и в режиме пакетной обработки (примерами могут служить различные версии ОС UNIX для суперкомпьютеров) или даже при различных сочетаниях всех трех режимов (например, ОС UNOS обеспечивает возможность одновременной работы как в режиме реального времени, так и в режиме разделения времени).

Следует отметить и такую особенность ОС UNIX, как мобильность, что позволяет легко переносить ее на компьютеры с различной архитектурой. Простота и удобство всех уровней интерфейса с пользователем (в данном случае пользователем является программист) обеспечивают прозрачность и ясность программ, создаваемых в среде ОС UNIX. Более того, широчайший набор имеющихся в среде ОС UNIX инструментальных

средств позволяет с их помощью решать широкий класс задач, вообще не прибегая к традиционному программированию.

Распространение ОС UNIX на мир персональных компьютеров в значительной степени обусловлено недостатками ОС MS-DOS, связанными с ее однопользовательской однопрограммной природой. Этому способствуют два обстоятельства. Во-первых, крупнейшие фирмы — разработчики пакетов прикладных программ (например, Ashton-Tate, Lotus и Samna International) перенесли свои наиболее популярные пакеты из среды ОС MS-DOS в среду ОС UNIX (точнее говоря, в среду ОС XENIX). Во-вторых, в среде различных версий ОС UNIX, предназначенных для работы на персональных компьютерах, имеются средства, обеспечивающие возможность выполнения программ, разработанных для среды ОС MS-DOS (к таким средствам относится, например, пакет VP/ix ОС XENIX и ОС DOS-Merge).

Следствием огромного интереса к ОС UNIX со стороны столь большого числа крупнейших фирм-разработчиков аппаратных и программных средств вычислительной техники явилось появление значительного числа не полностью совместимых друг с другом версий ОС UNIX. Уже в 1985 г. число таких версий перевалило за 70. Наличие столь большого числа не полностью совместимых друг с другом версий ОС UNIX вызвало серьезную озабоченность среди пользователей, которые по вполне понятным причинам хотят иметь единую унифицированную операционную среду, что невозможно без стандартизации ОС UNIX.

Большую работу по стандартизации ОС UNIX проводит основанная в конце 1984 г. Ассоциация производителей аппаратных и программных средств вычислительной техники X/OPEN. Весной 1985 г. эта Ассоциация подготовила документ X/OPEN Portability Guide, а к лету 1986 г. было продано уже свыше 7000 его копий. Второе, расширенное и дополненное издание этого документа появилось в 1987 г.

В 1988—1989 гг. к решению проблемы стандартизации ОС UNIX присоединились еще два участника: консорциум Open Software Foundation (OSF), созданный по инициативе фирм IBM и DEC, и консорциум UNIX International, организованный в противовес OSF по инициативе корпорации AT&T и фирмы Sun Microsystems. Появление этих двух новых участников внесло дополнительные трудности в процесс стандартизации, так как ознаменовало собой возникновение двух противостоящих группировок. Одну из них возглавляет фирма IBM, которой не дает покоя лидерство корпорации AT&T в вопросах, связанных с ОС UNIX, а вторую — корпорация AT&T, не желающая это лидерство уступать.

Говоря о стандартизации ОС UNIX, нельзя не упомянуть об успешно завершившейся стандартизации языка Си, для которого уже принят стандарт Американского национального института стандартов (ANSI).

Переходя к обсуждению перспектив развития ОС UNIX, необходимо отметить ряд обстоятельств. Во-первых, консорциум OSF объявил о разработке собственной версии ОС UNIX, которая получит название OSF—1. Во-вторых, корпорация AT&T, предполагающая в дальнейшем вместо языка Си использовать язык Си++, заявила о том, что новая версия System V Release 5 будет реализована на языке Си++ (этот язык, разработанный Б.Страуструпом, сотрудником фирмы Bell Laboratories, представляет собой дальнейшее развитие языка Си; язык

Си++ позволяет использовать объектно-ориентированный подход к программированию и абстрактные типы данных). В-третьих, по оценке американского журнала «Electronics», доли рынка, приходящиеся на ОС UNIX и MS-DOS, к началу 90-х гг. сравняются и каждая из них ориентировочно составит 22—23% от всего объема продаж.

Во второй половине 60-х гг. фирма Bell Laboratories была вовлечена в работы по проекту Multics, в которых принимали также участие фирма General Electric и Массачусетский технологический институт. В цели проекта Multics входило создание многопользовательской операционной системы разделения времени.

В 1969 г. фирма Bell Laboratories вышла из проекта Multics и полностью прекратила все связанные с этим проектом работы. Наиболее активное участие в проекте Multics со стороны фирмы Bell Laboratories принимали К.Томпсон, Д.М.Ритчи, М.Д.Микилрой, Дж.Ф.Осанна. Им не хотелось терять накопленные наработки, поэтому, несмотря на запрет руководства продолжать работы по созданию ОС, они начали искать различные способы использования полученных ими результатов. Например, ими рассматривались такие варианты, как разработка новой ОС для имевшегося в фирме Bell Laboratories компьютера GE—645 (для которого создавалась ОС Multics) и приобретение компьютера среднего класса (например, PDP—10 или Sigma—7) с целью разработки новой ОС для него. Параллельно с этим на имевшемся компьютере GE—645 осуществлялось моделирование файловой системы и схем управления памятью. Однако в силу ряда обстоятельств обе эти идеи были отвергнуты.

В это время К.Томпсон занимался разработкой программы Space Travel, моделировавшей движение основных небесных тел, входящих в состав Солнечной системы. Каждый прогон этой программы на компьютере GE—645 обходился в 75 долл., что было непомерно дорого. Поэтому К.Томпсон принялся за поиски какого-либо другого компьютера, использование которого обходилось бы не так дорого. В ходе своих поисков он обнаружил практически никем не используемый мини-компьютер PDP—7 — предшественник широко известных мини-компьютеров семейства PDP—11.

Тогда К.Томпсон начал перерабатывать свою программу Space Travel с тем, чтобы ее можно было использовать на мини-компьютере PDP—7. Однако оказалось, что переход к использованию мини-компьютера PDP—7 не так уж прост, поскольку имевшееся для него программное обеспечение состояло лишь из ассемблера и компоновщика. При этом необходимо было использовать перфоленту. Да и работать на этом мини-компьютере можно было лишь в монопольном режиме. Отсутствие программных средств, обеспечивающих удобство разработки, отладки и выполнения программ, натолкнуло К.Томпсона на мысль о создании соответствующих программных средств — так на новом витке возникла идея разработки собственной операционной системы.

В 1969 г. появилась написанная на языке Ассемблер однопользовательская ОС для мини-компьютера PDP—7, перенесенная через некоторое время на мини-компьютер PDP—9 (являвшийся также предшественником широко известного семейства мини-компьютеров PDP—11). Именно эта ОС и получила впоследствии название UNIX. Это название было предложено Б.У.Керниганом в 1970 г. и отражало противопоставление ОСистем (простая и ма-

ленькая ОС UNIX — сложная и большая ОС Multics). ОС UNIX заимствовала определенные черты у ОС Multics.

В 1970 г. по предложению Дж.Ф.Осанны фирма Bell Laboratories приобрела мини-компьютер PDP—11/20, который предполагалось использовать в проекте,. связанном с разработкой системы, облегчающей создание документации. В 1971 г. ОС UNIX была перенесена на мини-компьютер PDP—11/20 в качестве основы для системы ведения документации. В июне 1972 г. появилась новая версия ОС UNIX, также написанная на языке Ассемблер и известная впоследствии как версия 2.

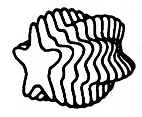
В это время К.Томпсон уже работал над созданием бестипового языка Би, строившегося на основе также бестипового языка Би-Си-Эл. В результате был создан новый язык — Эн-Би. Наконец, занимавшийся вопросами генерации кода для транслятора с языка Эн-Би Д.М.Ритчи создал на его основе получивший впоследствии широчайшую известность язык Си.

В 1973 г. впервые появилась версия 3. В мае 1975 г. создана первая пригодная для промышленной эксплуатации версия ОС UNIX, известная в настоящее время как версия 6. В декабре 1977 г. ОС UNIX была перенесена на компьютер Interdata 8/32. В результате такого переноса сформировалась еще одна версия ОС UNIX, в настоящее время известная как версия 7. В 1978 г. ОС UNIX была перенесена на микрокомпьютеры семейства VAX—11, эта версия получила название 32V.

После создания ОС UNIX в течение длительного времени она была известна лишь в узком кругу специалистов по операционным системам, поскольку использовалась лишь в фирме Bell Laboratories и вычислительных центрах ряда американских университетов. Затем последовал период многочисленных переносов ОС UNIX на компьютеры различных типов и создания ОС, подобных ОС UNIX. В истории создания и развития ОС UNIX особое место занимает 1980 г., когда ОС UNIX и подобные ей системы начали активно использоваться, в том числе и на персональных компьютерах. Именно в 1980 г. ОС UNIX привлекла к себе внимание делового мира — основного пользователя персональных компьютеров.

В этом же году корпорация AT&T (в которую входит фирма Bell Laboratories) выпустила версию ОС UNIX, получившую название System III, а в январе 1983 г. — версию System V. В январе 1984 г. корпорация AT&T выпустила версию ОС UNIX, получившую название System V Release 2.0, претендующую на то, чтобы стать стандартом.

Разработка ОС UNIX была высоко оценена специалистами. В 1982 г. К.Томпсон и Д.М.Ритчи за создание ОС UNIX были удостоены премии за достижения, ежегодно присуждаемой редакцией журнала Electronics. При этом необходимо отметить, что впервые за 9 лет с момента ее учреждения эта премия была вручена за разработку программного обеспечения, а не за разработку аппаратных средств. В 1983 г. за создание ОС UNIX К.Томпсон и Д.М.Ритчи были удостоены сразу двух премий Ассоциации специалистов по вычислительной технике: премии Тьюринга и премии за разработку нового программного обеспечения.



Весной 1985 г. корпорация AT&.T опубликовала документ, названный ею System V Interfage Definition (SVID), представляющий собой стандарт на ОС UNIX версии System V, а в 1986 г. — расширенное издание этого документа в трех томах. Деятельность по стандартизации ОС UNIX в настоящее время осуществляется рабочей группой Р 1003 IEEE, разработавшей стандарт Posix на мобильную ОС. Первый проект стандарта был подготовлен этой рабочей группой еще весной 1986 г.

М.И.БЕЛЯКОВ

Стандартизация UNIX

Широкое распространение, несомненные достоинства и хорошие перспективы операционной системы UNIX были начиная с 1984 г. поддержаны новым важным шагом — попытками стандартизовать ее пользовательские интерфейсы и язык Си, на котором система в основном написана.

Существенный прогресс к настоящему моменту достигнут в этом направлении следующими организациями:

— комитетом ANSI X3L11 Американского национального института стандартов, разработавшим стандарт языка Си с учетом системных библиотек;

— комитетом IEEE P1003, разрабатывающим стандарт POSIX на пользовательские интерфейсы системы;

— группой X/Open, объединившей с целью выработки такого стандарта ряд западноевропейских, американских и японских компаний.

Американский национальный стандарт языка Си разработан Техническим комитетом X3J11 и адресован как программистам, использующим Си, так и создателям конкретных реализаций языка. Цель разработки стандарта состоит в том, чтобы дать недвусмысленное и машинно-независимое описание языка Си. Поскольку в языке нет изобразительных средств для ввода-вывода и многих других типовых операций, стандарт включает достаточно содержательную библиотеку функций как неотъемлемую часть языка. В библиотеке, в частности, имеются средства локализации, т.е. учета национального представления различных величин и обозначений.

Проект стандарта интерфейсов мобильной операционной среды, названный POSIX, подготовлен рабочими группами американского института IEEE. Название мобильной системы — POSIX — является товарным знаком IEEE. В стандарте воплощена точка зрения на чисто мобильный интерфейс программ с операционной системой.

К настоящему моменту выпущена окончательная версия проекта стандарта POSIX на общие функции программного интерфейса. Другие документы POSIX еще требуют доработки. Программный интерфейс POSIX реализуется как в операционных системах, совместимых с UNIX, так и в системах, построенных на других концепциях.

В состав материалов POSIX входит общее руководство по стандартам POSIX, в частности описание системной архитектуры POSIX. Наиболее проработан общий программный интерфейс POSIX. От традиционных вещей общего программного интерфейса UNIX документ отличается специальными функциями обработки сигналов, специальными функциями работы с терминалом и рядом мелких особенностей.

Основой командного языка в POSIX является язык интерпретатора Bourne shell, старейшего из командных интерпретаторов UNIX. Сделаны расширения в духе двух других наиболее известных интерпретаторов C shell и Korn shell. Введена воз-

можность редактирования текущей командной строки или одной из предыдущих командных строк в стиле одного из текстовых редакторов UNIX. В командный интерфейс входит система извлечения прикладных программ из дистрибутивного носителя, установки их в требуемых местах системы, поддержки базы данных, доступной пользователям и прикладным программам и используемой для отображения логических ресурсов программы на реальные ресурсы системы.

В материалах POSIX сделана начальная проработка интерфейса реального времени. Имеются две части этого стандарта: часть «a», посвященная понятию параллелизма внутри процесса (Thread), и часть «b», посвященная расширениям для целей реального времени. Реальное время в операционных системах определено в стандарте как способность операционной системы обеспечить требуемый уровень сервиса в условиях, когда время реакции ограничено. Стандарт предусматривает многие функциональные возможности для реализации реального времени.

1. Двоичные семафоры — минимум примитивов синхронизации как основа для построения более сложных средств синхронизации в прикладных программах; семафор является новым типом специального файла; операции: установить, ждать, ждать условно.

2. Блокировка процессов в памяти — для устранения системных издержек при считывании процесса из памяти второго уровня.

3. Разделяемая память — одновременный доступ прикладных программ к общему участку их образов памяти; разделяемая память выступает как еще один тип специального файла. Допустимы общие операции работы с файлами типа открытия-закрытия, а кроме того, вводятся специальные процедуры, связанные отображением разделяемой области в адресное пространство процесса. Ряд типовых файловых функций из программного интерфейса POSIX (смена владельца, кода защиты, работа с именами связей и т.п.) допустим для специального файла разделяемой памяти.

4. Приоритетное планирование:

— допускается до 32 приоритетов, для каждого приоритета свой список процессов;

— для каждой дисциплины планирования свой диапазон приоритетов, который может вырождаться либо до одного приоритета, либо до полного диапазона приоритетов; дисциплина означает определенные правила включения в список и упорядочения списка;

— независимо от дисциплины при выборе нового процесса для занятия процессора берется процесс из головы списка с наибольшим приоритетом.

5. Допустимы три дисциплины: FIFO (в порядке активизации), round robin FIFO (с квантованием времени) и, по крайней мере, еще одна дисциплина; в случае FIFO список упорядочивается по времени активизации процессов; в случае квантования список упорядочивается, как при FIFO, но есть дополнительная причина для снятия процесса с процессора (исчерпание кванта); стандарт требует, чтобы реализация включала еще хотя бы одну дополнительную дисциплину.

6. Асинхронные прерывания — средства, заменяющие механизм сигналов, не совместимые с ним и позволяющие организовывать очередь асинхронных прерываний.

7. Таймеры — два типа таймеров: срабатывающий один раз и срабатывающий периодически; задаваемое время может быть абсолютным и относительным (к текущему моменту).

8. Межпроцессное (внесетевое) взаимодействие.

9. Синхронный ввод-вывод — более детальные (чем имеющиеся сейчас) процедуры синхронного ввода-вывода, обеспечивающие контроль целостности данных и файлов со стороны прикладной программы.

10. Асинхронный ввод-вывод.

11. Файлы реального времени.

Пока в проекте стандарта для расширения реального времени описаны только семафоры, разделяемая память, приоритетное планирование, таймеры и синхронный ввод-вывод.

Понятие «thread» является новым для UNIX и обозначает единицу последовательного выполнения внутри процесса. Назовем его подпроцессом, хотя вполне возможны другие переводы. Каждый подпроцесс имеет свой идентификатор (в пределах процесса), приоритет и дисциплину планирования, таймеры, код ошибки (переменную errno), системные ресурсы, необходимые для функционирования. Подпроцессы функционируют внутри единого адресного пространства процесса.

Подпроцессы могут быть использованы для параллелизма, присущего многооконному интерфейсу, обработке событий в режиме реального времени, задачам языка Ада, обработке транзакций, сетевым и распределенным системам. Процессы взаимодействуют между собой независимо от их внутреннего строения. Реализован ли процесс в виде единой программы или является совокупностью подпроцессов — этот факт не влияет на окружение процесса. Подпроцессы прежде всего нужны там, где цена создания нового параллельного действия соизмерима с ценой выполнения вызова функции. Поэтому подпроцессы должны создаваться максимально быстро.

В стандарте рассматриваются:

— средства синхронизации подпроцессов, обеспечивающие взаимное исключение; используются специальные синхронизирующие объекты, называемые mutual exclusion, и условные переменные; механизм синхронизации напоминает исключение с помощью критических участков;

— различные способы завершения подпроцесса;

— работа традиционных функций управления процессами и сигналами в случае подпроцессной структуры процессов;

— проблемы повторной входимости в случае подпроцессной структуры процессов.

Интерфейс защиты (управляемый доступ) разработан в POSIX достаточно глубоко и резко отличается от традиционного подхода UNIX к этой проблеме. Объекты, на которые распространяется управляемый доступ, — файлы всех типов (включая программные каналы) и процессы. Файлы выступают в роли объектов, процессы — в роли субъектов. Различается доступ к объектам, основанный на дискретном выборе, и доступ, основанный на полномочиях.

Дискретный выбор подразумевает соответствие списков, когда для каждого объекта есть список субъектов, доступ которых к объекту разрушен. Таковым является механизм доступа в имеющихся версиях UNIX, и он достаточен для многих применений.

Доступ, основанный на полномочиях, использует соответствие меток. Вводятся метки объектов и субъектов, т.е. процессов и файлов. Вводятся понятия доминанты и равенства меток (как отношения между метками). Создаваемый файл наследует метку от создавшего его процесса. Вводятся соотношения, определяющие права процессов по отношению к файлам.

Несмотря на то что дискретный доступ соответствует по философии механизму защиты, принятому в UNIX, интерфейс его резко отличается от используемого. Вводится существенная детализация механизма и большое количество новых понятий, функций и команд, реализующих этот механизм. Вводимые средства можно разделить на такие группы:

— для работы со списками доступа при дискретной защите;

— для проверки права доступа;

— для управления защитой на основе полномочий;
— связанные с привилегиями.

В рамках POSIX начались работы над интерфейсом системного администратора. Работы предполагалось завершить к 1993 г. К проблеме системного администратора в POSIX подходят с объектно-ориентированных позиций. Предполагается определить объекты, имеющие интерес с точки зрения системы, удовлетворяющей спецификациям POSIX, а затем определить множества действий, которые можно выполнять над такими объектами. В качестве классов предполагаются пользователь, группа пользователей, устройство, файловая система, процесс, очередь, вход в очередь, машина, система, администратор, программное обеспечение и пр. Определяются атрибуты таких классов, операции над классами и события, которые могут с ними происходить.

Подобный подход, безусловно изящный с точки зрения теории, по-видимому, не просто сделать удобным на практике, учитывая, что работа системного администратора уже имеет свои наработанные годами традиции. Впрочем, у рабочей группы есть довольно много способов для реализации своих идей, так что пока рано говорить о том, каков будет интерфейс системного администратора.

Сетевым интерфейсам посвящен проект стандарта, состоящий из нескольких частей. Одна часть стандарта характеризует доступ к файлам в сетевой среде и состоит из трех секций: секция, описывающая программный интерфейс независимо от языка программирования; секция, зависящая от используемого языка программирования; секция, описывающая командный интерфейс доступа к файлам. Первая секция получена извлечением из P1003.1 всего того, что относится к доступу к файлам. Затем поведение функций в нераспределенной среде дополняется описанием их поведения в сетевом варианте.

Вторая часть сетевого стандарта посвящена межпроцессному взаимодействию в сетевой среде. Взаимодействие процессов в сетевой среде рассматривается как расширение программного интерфейса POSIX. Рассматриваются следующие вопросы:

— установка и разрыв соединения;
— передача данных при межпроцессном взаимодействии;
— управление событиями;
— функционирование сети;
— средства, не зависящие от сетевого протокола;
— средства, зависящие от сетевого протокола.

В рамках POSIX рассматривается связь с программами на Фортране 77. Прежде всего дается фортранное определение констант и макросов, используемых в программном интерфейсе POSIX. Поскольку этот интерфейс основан на структурах данных языка Си, возникает проблема доступа к ним из фортранных программ. Решение сделано в духе абстрактных типов данных, представляющих требуемые агрегатные структуры языка Си. Вводятся специальные фортранные подпрограммы (subroutine) для манипуляции с такими структурами. Дается фортранное изображение ряда примитивных функций из общего программного интерфейса POSIX: fork, exec, wait и т.п.

Целью проекта X/Open является создание среды прикладного программирования (common applications environment — CAE) для функционирования мобильных прикладных программ, которые без модификаций или с минимальными модификациями могут быть перенесены с одной машины на другую. Основная философия стандартизации в рамках X/Open: не создавать новые стандарты, а использовать имеющиеся. В этом основное отличие проекта X/Open от других проектов и причина быстрого продвижения проекта по степени охвата различных интерфейсов, включая интерфейсы с языками программирования, средства управления данными, межпроцессное и сетевое взаимодействие. Однако этот факт определяет несамосто-

ятельность группы и ее зависимость от фактического и юридического статуса других проектов.

Первоначально предложения группы X/Open основывались на спецификации фирмы AT&T для разработанной ею UNIX System V и известной под названием SVID (System V Interface Definition). Выпущенная в декабре 1988 г. третья версия руководства опирается на стандарт языка Си и программный интерфейс POSIX. Проблемы интернационализации разработаны в предложениях X/Open достаточно глубоко. Поддержка национального языка подразумевает:

— 8-битное кодирование символов;

— раздельное существование программы и выдаваемых ею сообщений (сообщения переводятся на требуемый язык и извлекаются во время работы программы);

— механизм объявления национального языка и национальных обозначений;

— специальная библиотека функций, ориентированных на поддержку национальных языков, в том числе динамическое определение специфических для данного языка данных;

— регулярные выражения, ориентированные на поддержку национальных языков;

— специальные команды, ориентированные на поддержку национальных языков.

Описание Кобола опирается на ANSI стандарт Кобола 1985 г. и его международный эквивалент. Описание Фортрана опирается на его ANSI стандарт 1978 г., описание Паскаля — на международный стандарт 1983 г., описание языка запросов SOL — на его ANSI стандарт 1986 г., описание последовательного метода доступа ISAM — на описание фирмы Informix Corporation. Многооконный интерфейс соответствует X Window. Механизм межпроцессного взаимодействия в предложениях X/Open включает в себя семафоры, разделяемую память и посылку сообщений. Интерфейс семафоров и разделяемой памяти отличен от этих механизмов в POSIX и соответствует их реализации в UNIX System V.

Стандартизация интерфейсов операционной системы является беспрецедентным шагом, и появление юридического стандарта поднимет, как ожидается, и без того высокие акции UNIX.

ЧИТАТЕЛЮ НА ЗАМЕТКУ

НАВИГАЦИОННАЯ СИСТЕМА НА БАЗЕ ЭВМ

Разработан оптимальный адаптивный регулятор инерциальной навигационной системы, основанный на минимизации суммы дисперсий ошибок элементов всей системы. Этот регулятор позволяет проводить автоматическую компенсацию ошибок на длительных интервалах функционирования системы, что чрезвычайно важно для практических приложений (в авиации, космонавтике, мореходном деле). Показано, что качество автоматического управления по сигналам подобной инерциальной системы повышается за счет увеличения степени наблюдаемости ее ошибок.

Журнал «Известия РАН», серия «Техническая кибернетика». — 1992. — № 2. — С. 178—183.

М.М.ДЕНИСОВ, Д.И.ТОРОПОВ

Компьютеры U-СЕРИИ

В последнее время в мире возрастает интерес к операционной системе (ОС) UNIX. Открытая реализация UNIX и OSI-возможностей используется для взаимодействия и обмена информацией между системами различных поставщиков. Ряд международных организаций, в частности UNIX International, работают в области внедрения UNIX — систем и стандартов на ОС UNIX.

Компьютеры U-серии производит корпорация Unisys, являющаяся крупнейшим в Европе поставщиком машин этого типа. При реализации своих систем она стремится к возможно большей их преемственности, открытости и соответствию стандартам. Unisys в значительной степени основывается на концепции Взаимодействия Открытых Систем и стандарт на ОС UNIX.

Автоматизация деятельности небольших подразделений требует, как правило, индивидуальных решений, которые должны без больших дополнительных затрат адаптироваться к постоянно изменяющимся требованиям жизни. Секретариаты, плановые отделы, бухгалтерии, отделы сбыта и снабжения, отделы разработчиков, программистов и т.д. нуждаются в дополнительных вычислительных системах, не требующих специальных помещений и способных удовлетворить разнообразные коллективные и индивидуальные требования. Обмен данными с ЭВМ различных служб и уровней, способность к расширению и модификации также необходимы для перспективных систем.

UNIX — ориентированные компьютеры U-серии образуют возрастающий по мощности ряд ЭВМ от настольных рабочих станций до многопроцессорных супермини, обладающих широкими коммуникационными и сетевыми возможностями для связи с удаленными терминалами, рабочими станциями и с HOST-компьютерами.

Стандартизованная версия операционной системы UNIX — Unisys System V делает эти ЭВМ совместимыми по программному обеспечению между собой и другими UNIX — ориентированными системами.

Широкий спектр возрастающих по мощности компьютеров U-серии и уникальный набор программного обеспечения дают возможность Unisys построить расширенную платформу VAP.

Value Added Platform (VAP)

Понятие value added широко используется в англоязычной литературе. Value added означает, что к исходному продукту добавляется нечто, что может быть расширением возможностей применения исходного продукта, или некий набор услуг, или что-то другое, что увеличивает потребительскую ценность исходного продукта. Далее мы будем употреблять словосочетание «расширенная платформа».

Расширенная платформа VAP — это концепция, включающая аппаратуру, программное обеспечение и обслуживание пользователей, разработанная для обеспечения открытыми UNIX-системами как больших, так и малых пользователей. В этой концепции Unisys объединяет свои обязательства по поддержке открытых систем и лидерство в области разработки UNIX-продукции. Компоненты Value Added Platform могут быть разделены по пяти уровням. Как показано на рисунке, все компьютеры U-серии работают под управлением ОС UNIX Cystem V (или Unisys System V), которая поддерживает ряд системных программных продуктов, представленных в виде пяти ключевых направлений использования вычислительных мощностей.

ВЕРТИКАЛЬНЫЕ ПРИЛОЖЕНИЯ (Image, Line of Business)

ГОРИЗОНТАЛЬНЫЕ ПРИЛОЖЕНИЯ (OFIS, AL, EaDi)

Операционная среда 1 — 0	Инструменты разработки программ	Управление данными (СУБД)	Коммуникации и сети	Интерактивная обработка запросов

Операционная система UNIX SYSTEM V

АППАРАТУРА	U-Series

ВЕРТИКАЛЬНЫЕ ПРИЛОЖЕНИЯ (Image, Line of Business)

Производимые в настоящее время UNIX-компьютеры Unisys обладают разнообразными характеристиками по объемам памяти и производительности, что дает пользователю возможность выбора подходящей модификации.

Над системным ПО расположены разработанные Unisys горизонтальные приложения, которые могут использоваться для объединения средств системного уровня между собой и со следующим уровнем вертикальных приложений. Вертикальные приложения могут быть выбраны потребителем из множества специализированных программных пакетов (Unisys или других поставщиков), работающих в среде UNIX System V.

Все ЭВМ U-серии работают в обычных климатических условиях рабочего помещения (офиса) и не требуют специального сетевого питания и специальных условий по температуре и влажности.

Операционная система Unisys System V

Unisys System Operating System — это полностью сертифицированный продукт, развитие стандартной операционной системы UNIX System V Release 3.2 корпорации AT&T. Unisys System V включает все стандартные возможности UNIX, совместима с промышленными стандартами:

— FIPS 151.1 NIST;

— 1EEE 1003.1 POSIX;

— X/Open XPG2,

а также расширена за счет включения таких средств, как:

— встроенная поддержка разнообразных национальных языков MNLS (Multi-National Language Supplement);

— транспортный интерфейс TLI (Transport Layer Interface) в соответствии с моделью OSI/ISO;

— сетевая файловая система NES (Network File System), обеспечивающая прозрачный сетевой доступ к файлам, расположенным на различных компьютерах с различными операционными системами;

— меню-ориентированные пользовательские интерфейсы с операционной системой и утилитами;
— средства поддержки программ реального времени;
— поддержка до 4 Гбайт виртуального адресного пространства на одну задачу.

Программы операционной среды реализуют административные системные функции и включают:
— меню-доступ к системным функциям SA Menus;
— средства авторизации доступа Security Tools;
— средства поддержки национальных языков MNLS;
— средства оконной работы для одновременного обзора данных от разных источников MINDOWS.

U-серия — платформа для рывка в будущее

Множество приложений, обслуживающих специфические области деятельности, уже существует в виде типовых адаптивных пакетов. Их поставляет как Unisys, так и ряд других независимых программистских фирм. Тем не менее конечный пользователь без особого труда с помощью мощных средств автоматизации программирования, таких, как 4GL, может самостоятельно или пользуясь консультациями специалистов разработать собственный пакет. Существующие программные средства включают:
— видеообработку документов;
— управление производством;
— обработку штрих-кодов;
— управление финансами;
— библиотечное дело и т.д.

Следует отметить, что библиотека программных средств Unisys для горизонтальных и вертикальных приложений исключительно богата. Организациям, которые заботятся о рентабельности и быстром вводе в эксплуатацию прикладных систем, значительно выгоднее использовать уже готовые пакеты, чем разрабатывать с нуля новые.

Unisys с помощью Расширенной Платформы VAP обеспечивает пользователей существенными преимуществами открытых систем и сетей:
— широким спектром вычислителей;
— множеством системных программных компонент;
— гарантиями и удобством единственного поставщика;
— сервисом и поддержкой по всему миру;
— взаимозаменяемостью аппаратуры и переносимостью программ;
— своевременными улучшениями и расширениями.

Выбирая компьютер U-серии, вы приобретаете на многие годы гибкий мощный инструмент увеличения производительности и прибыльности своего дела.

ДЛЯ СРАВНЕНИЯ

OS/2	DOS	OC UNIX
• 32-разрядная система • многозадачная система • быстрый перенос прикладных программ DOS	• недорогая система • есть много прикладных программ • сложившийся рынок	• 32-разрядная система • независимость от вида базовой машины • многозадачная, многопользовательская система



Для создания прикладных систем с графическими пользовательскими интерфейсами используют различные программно-аппаратные платформы. Одной из них является оконная система NeWS фирмы Sun Microsystems на UNIX-рабочих станциях.

А.В.КАШИРИН, В.А.НИКОЛАЕВ

Языковые средства NeWS

Введение

В ряду подобных систем (Sun Windows, XWindow) появление NeWS было достаточно ярким событием, которое до сих пор фактически не отражено в отечественной литературе. Более того, как показала практика, при необходимости непросто получить содержательную информацию и из доступных зарубежных изданий.

NeWS представляет интерес не только как средство решения текущих практических задач. Авторы этой системы в первую очередь рассматривали ее как попытку утвердить новый перспективный подход к разработке графических интерфейсов. Способы, которыми здесь было достигнуто сочетание графики, переносимости (имеются в виду реализации NeWS для OS/2 и Macintosh), независимости от характеристик устройств ввода и отображения, сетевого сервиса, несомненно, представляют интерес для исследователей и разработчиков.

В этой статье дано общее представление о структуре и функционировании NeWS. Основное внимание сосредоточено на характеристике языковых средств разработки прикладных программ с графическими интерфейсами на базе NeWS.

Наряду с небольшой частью доступной документации по NeWS фирмы Sun Microsystems основным источником информации для нас было изучение демонстрационных приложений и опыта практического применения реализации NeWS на персональной рабочей станции Personal Iris фирмы Silicon Graphics.

NeWS — торговая марка Sun Microsystems, Inc., PostScript, Display PostScript — торговые марки Adobe Systems, Inc.; UNIX — зарегистрированный торговый знак AT&T; X Window System — торговая марка Massachusetts Institute of Technology; Personal IRIS — торговая марка Silicon Grafics, Inc.; NeXT — торговая марка NeXT, Inc.

Особенности NeWS-подхода к созданию графических интерфейсов

Дословно NeWS расшифровывается как сетевая расширяемая оконная система (Network Extensible Window System). Аналогично системе X Window архитектура NeWS основана на модели «клиент-сервер». Клиенты соответствуют прикладной части сетевого ПО, а сервер — системной части, инвариантной к приложениям, предоставляющей клиентам оконный сервис. Клиенты и сервер реализуются отдельными процессами операционной системы. Взаимодействие клиента с сервером осуществляется путем передачи сообщений через коммуникационную среду.

Особенностью NeWS является механизм расширения функций сервера, на основе которого строится все взаимодействие клиентов с NeWS-сервером.

Несмотря на присутствие слова «оконный» в названии системы, на уровне базовых функций NeWS нет какой-либо фиксации стиля пользовательских интерфейсов, конструкции окон, меню и прочих элементов, которые видит на экране и которыми манипулирует пользователь. NeWS-сервер имеет примитивные механизмы для их реализации. Настройка же сервера на обеспечение требуемого в приложении развитого оконного сервиса выполняется самим клиентом путем расширения базовых функций сервера. Пользоваться этими расширениями могут и другие клиенты.

Основную идею NeWS можно выразить следующим образом. Если в других оконных системах клиент взаимодействует с сервером через ограниченный набор типов сообщений, то каждое сообщение клиента NeWS-серверу — это программа на алгоритмически полном интерпретируемом языке программирования. Со стороны клиента NeWS-сервер выглядит как интерпретатор этого языка. При наличии в языке средств определения новых понятий сервер автоматически становится расширяемой системой. Если же в ядро языка будут встроены примитивы, достаточные для описания конструкций и алгоритмов функционирования оконных графических интерфейсов, то мы получим расширяемую оконную систему.

Таким образом, базовые функции NeWS-сервера, возможности их расширения, гибкость пользовательских интерфейсов на базе NeWS определяются выбором языка общения клиентов с сервером. В качестве этого языка был взят язык PostScript фирмы Adobe Systems, ядро которого было расширено дополнительными возможностями (в данной работе этот вариант PostScript будет называться NeWS PostScript).

Показанный способ развития модели «клиент-сервер» имеет прямое влияние на эффективность удаленного доступа к серверу. При решении многих задач за счет такой стратегии существенно уменьшается интенсивность взаимодействий клиента и сервера. Например, если требуется нарисовать координатную сетку на экране, то нет необходимости в цикле передавать множество линий в оконную систему. Достаточно один раз передать программу генерации полного изображения сети.

Можно привести много примеров из классов задач, требующих возможности визуального оживления, например изменение формы некоторого графического объекта по заранее известному закону ее изменения в зависимости от текущего положения курсора. В данном случае клиент, находящийся в любой точке сети, избавляется от необходимости отслеживать движение курсора, поскольку эта функция возлагается на сервер.

Возможность одновременного обслуживания нескольких клиентов обеспечивается в NeWS-сервере следующими решениями. Вся работа сервера протекает в рамках одного процесса операционной системы. Однако этот процесс имеет свою внутреннюю структуру, элементами которой являются квазипараллельные подпроцессы, называемые lightweight-процессами (в литературе в таком же значении часто используют термин «thread»). Каждому клиенту (UNIX-процесс) при подключении к серверу ставится в соответствие один lightweight-процесс, который будет интерпретировать сообщения данного клиента. В ходе интерпретации этот процесс может исполнять операции с дисплеем, реагировать на клавиатуру и мышь, взаимодействовать с другими lightweight-процессами. Для построения программ таких процессов NeWS PostScript предлагает удобную и эффективно реализуемую систему понятий.

В следующих разделах дана общая характеристика языка PostScript и его расширений в NeWS. Эта характеристика дает представление о базовых функциональных возможностях NeWS-сервера. На практике же сразу после инициализации он предоставляет клиентам существенно более развитый сервис, реализуемый как расширение базовых функций. Эти расширения определяются в целом ряде библиотечных NeWS PostScript программ, выполняемых в процессе инициализации.

Далее будет показано, каким образом должны строиться программы NeWS-клиентов. Сейчас же кратко опишем вспомогательные средства для разработки этих программ на языке Си. Такие средства входят в стандартный состав NeWS.

Язык PostScript

PostScript является примером языка, в котором мощные графические средства встроены в рамки легко расширяемого интерпретируемого языка программирования с простым синтаксисом, универсальной моделью данных и широким спектром возможностей для управления выполнением программ.

Появление языка PostScript (1982 г.) было вызвано потребностью в средствах описания процессов автоматического синтеза двумерных тексто-графических изображений на растровых устройствах

вывода информации. В таком качестве он получил достаточно широкое признание. Целый ряд принтеров аппаратно интерпретирует описания документов на языке PostScript.

Широкое распространение получили так называемые настольные издательские системы, позволяющие интерактивно конструировать графические документы на экране монитора, изготавливать их твердые копии, передавать их PostScript-описания в другие системы.

Модель синтеза изображений

Успех PostScript во многом был предопределен свойствами принятой за основу модели синтеза изображений (imaging model). Привлекательность этой модели послужила одной из главных причин использования языка PostScript не только в NeWS, но и в других системах (например, язык Display PostScript на NeXT компьютерах).

Представления о формировании изображений, реализованные в языке PostScript, отличаются простотой, полнотой и абстрактностью. В основе лежит stencil/paint-модель, что можно перевести как «раскраска через трафарет». Под трафаретом (stencil) здесь понимается некоторый контур, специфицируемый бесконечно тонкой границей, составленной из гладких кривых в непрерывном координатном пространстве. Раскраска (paint) — это нанесение либо краски одного цвета, либо любого изображения на поверхность изображения. Предполагается, что эта операция всегда выполняется через трафарет. Результирующее изображение соответственно будет иметь контуры, определяемые выбором трафарета.

Такая модель полна, поскольку и линии, и текстовые символы могут быть определены через трафареты. Она проста для понимания вследствие удачного выбора метафоры. Абстрактность этой модели выражается, например, в том, что определение фонта допускает его различные реализации: либо как битовые карты, либо как векторный или гладкий контур. За счет абстрактности модели синтеза изображений PostScript приобрел свойство независимости от характеристик устройств отображения. Эволюция аппаратных средств визуализации не является препятствием для использования и распространения этого языка.

PostScript-программист выражает процесс построения желаемой картины с помощью графических примитивов (операторов) языка.

Трафарет в языке PostScript специфируется произвольными комбинациями векторов, дуг и кривых высокого порядка с помощью операторов конструирования траекторий (path construction operators). Траектории определяются в терминах точек, специфицируемых координатами на так называемых страницах (page). По умолчанию используется координатная система PostStript, которая может быть изменена операторами линейных трансформаций координатного пространства (coordinates system operators), выполняющими сдвиг начала координат, изменение масштаба, вращение. Построение изображений на странице осуществляется операторами раскраски (painting operators). Построенная страница воспроизводится на устройстве отображения оператором showpage.

Операторы PostScript, как правило, требуют нескольких аргументов, часть из них задается явно, другие же — неявно, как побочный эффект ранее выполненных операторов установки графического состояния (graphics state operators). Среди таких неявных аргументов наиболее существенными являются current page (текущая страница), current path (определяет текущий трафарет), current clipping path (определяет границу рабочей области текущей страницы), current transformation matrix (матрица, задающая отображение координатного пространства пользователя в координатное пространство устройства отображения), current font (текущий фонт, определяющий графические представления текстовых символов). Кроме того, в графическое состояние входят текущее значение цвета, толщина линий, текущее положение в координатном пространстве и другие. В языке предусмотрены операторы для сохранения и восстановления графических состояний.

Учитывая специфику структур данных и механизмов работы с описаниями символов и фонтов, соответствующие

операторы выделены в PostScript в отдельную группу (character and font operators). Эти операторы позволяют описывать, выбирать и модифицировать фонты.

Для полноты представления о графических возможностях PostScript перечислим основные операторы из группы painting-операторов: fill (закрашивает область страницы, задаваемую текущей траекторией), stroke (вырисовывает линию по текущей траектории), show (специальный оператор для вывода изображения символа из текущего фонта), image (помещает на текущую страницу дискретное изображение, полученное либо из внешнего источника, либо синтезированного программно), imagemask (получает изображение аналогично image, но использует его как маску для закраски страницы текущим цветом).

Характеристика синтаксиса и модели вычислений

Наряду с возможностями графических построений PostScript включает в себя как специфичные, так и обычные для языков программирования возможности. Отдельные идеи и элементы PostScript заимствованы из других широко известных языков.

Синтаксис этого языка очень похож на синтаксис языка Forth. Выражения языка записываются в постфиксной нотации, в которой операнды предшествуют операторам. PostScript содержит большое число встроенных операторов, но имена их не зарезервированы и могут быть переопределены программистом. Количество специальных символов очень мало.

К отличительной особенности PostScript относится отсутствие какого-либо синтаксического оформления понятия программы. PostScript-интерпретатор не считывает программу целиком перед ее выполнением, а обрабатывает ее последовательно. Из последовательности символов текстового представления программы (входного потока) по синтаксическим правилам формируются лексемы (tokens), из одной или более лексем создается PostScript-объект (значение данных в памяти интерпретатора), и затем этот объект сразу выполняется. Далее этот процесс повторяется до конца входного потока. При этом, конечно, выполнение программы может иметь побочные эффекты. В частности, они могут выражаться в создании процедурных объектов в памяти интерпретатора, предназначенных для последующих вызовов.

Очевидно, что такой синтаксис и организация выполнения программ очень подходят для использования PostScript в системе NeWS, где программа формируется последовательно во времени как поток сообщений от клиента к серверу.

Модель данных языка включает в себя обычный набор типов данных (числа, строки, массивы, булевы значения, файлы). Одним из основных понятий PostScript является словарь (dictionary) — ассоциативная таблица, к элементам которой относятся пары «имя—значение». Наряду с перечисленными типами объектами манипуляций и передач в PostScript-программе могут быть и те элементы, которые обычно рассматриваются как элементы программ (имена, операторы, процедуры). В этом языке любой объект может трактоваться как данные и может быть выполнен как часть программы. За счет этого PostScript приобрел свойства, характерные для языков программирования символьных вычислений, таких, как Lisp.

Каждый PostScript-объект имеет скрытую структуру из трех элементов: тип, атрибуты (литеральный/выполнимый, право доступа) и значение. Эти элементы объекта относительно независимы. Например, два объекта могут иметь один и тот же тип, разделять одно значение, но иметь разные значения атрибутов). Текущее состояние элементов объекта определяет семантику его выполнения.

В основу модели вычислений языка PostScript положены принципы вычислений на стеках. Для представления состояния выполнения программ PostScript-интерпретатор использует четыре различных стека — стек операндов (operand stack), стек словарей (dictionary stack), стек вызовов (execution stack) и стек графических состояний (graphics state stack).

Стек операндов используется для передачи операндов PostScript-операторам и для размещения результатов выполнения операторов. В этот стек могут быть

положены любые PostScript-объекты. Контроль типов операндов осуществляется динамически для всех встроенных операторов.

Стек словарей определяет текущий контекст для неявного поиска значений выполнимых имен. Поиск ведется сверху вниз по всем словарям стека до первого определения данного имени. Таким образом, на основе стека словарей реализуется модель динамического связывания имен и значений.

Верхний элемент стека словарей называется текущим словарем (current dictionary). Как правило, именно в этот словарь помещаются определения имен (пары «имя—значение») в ходе выполнения программы. Однако есть возможности для размещения новых определений в словари непосредственно через их имена. Указанные возможности задания новых определений и переопределения ранее установленных значений, а также операторы изменения стека словарей делают PostScript легко расширяемым языком, а PostScript-программы — очень гибкими. Эти свойства хорошо соответствовали целям разработчиков системы NeWS и, видимо, являлись одной из главных причин выбора PostScript.

На стеке вызовов сохраняются выполнимые объекты (процедуры и файлы). Когда интерпретатор прерывает выполнение одного объекта для выполнения другого, то он кладет прерванный объект в этот стек, а когда завершает выполнение объекта, то вынимает верхний объект из стека и продолжает его выполнение.

Как должно быть ясно из сказанного, графическое состояние имеет сложную структуру, состоящую из множества объектов. Текущие значения этих объектов определяют контекст выполнения графических операторов в текущей точке выполнения программы. PostScript-программа может сохранять и восстанавливать графические состояния с помощью двух операторов — gsave и grestore. Для этого и используется стек графических состояний.

Упомянутые стеки полностью независимы, и доступ к ним осуществляется различными средствами. Стек операндов и стек словарей управляются непосредственно выполняемой программой. Наоборот, стеком вызовов полностью управляет интерпретатор, а из программы он может только читаться, но не модифицироваться. Доступ к этим стекам и стеку графических состояний поддерживается соответствующими операторами языка.

В целом возможности PostScript как языка общего назначения определяют своей семантикой встроенные операторы следующих типов.

1. Операторы манипулирования стеком операндов.
2. Операторы для работы со словарями и стеком словарей.
3. Операторы манипулирования строками и массивами.
4. Арифметические и математические операторы.
5. Операторы сравнения и логические операторы.
6. Операторы изменения и анализа атрибутов объектов, приведения типов объектов.
7. Операторы управления вычислениями.
8. Операторы доступа к файлам.

Расширения языка PostScript в NeWS

В силу иного предназначения язык PostScript оказывается функционально недостаточным для его использования в качестве языка общения клиентов с NeWS-сервером.

Структура NeWS предполагает, что средствами такого языка будут выражаться все аспекты организации функционирования пользовательских прикладных систем. Соответственно дополнительно к вопросам синтеза и воспроизведения тексто-графических изображений перед разработчиком встают задачи визуального оживления этих изображений, задания требуемой чувствительности его элементов к воздействиям пользователя, задания требуемого способа взаимодействия асинхронных клиентов NeWS-сервера.

Только на основе имеющихся встроенных возможностей языка PostScript задачи таких типов не решить. Для этих целей в NeWS PostScript были введены дополнительные понятия, реализуемые через новые типы объектов, и соответствующие новые встроенные операторы. К числу основных дополнительных понятий относятся значения следующих тер-

минов этого языка: canvas (холст), process (процесс), event (событие) и interest (интерес). Их содержание раскрыто далее.

Кроме того, на уровень модели данных языка были вынесены объекты, представляющие цвет, графическое состояние и траектории. Для манипулирования этими объектами введены соответствующие типы операторов.

Холсты

Холст замещает понятие страницы стандартного PostScript и трактуется как поверхность, на которой строится изображение обычным набором графических примитивов этого языка. Окна, меню, pop-up-сообщения, различные элементы управления в графических интерфейсах, реализуемых на базе NeWS, — все основываются на холстах.

Выполнение NeWS PostScript-программы может приводить к созданию множества холстов. Все холсты, параллельно существующие в NeWS-сервере, являются элементами одной иерархической структуры. Ее корень соответствует физическому экрану и имеет имя framebuffer. Этот холст создается оператором createdevice при инициализации NeWS-сервера.

Каждый новый холст создается на некотором уже существующем, который становится его предком. Посредством ссылок на предка и формируется дерево холстов. Дополнительно к связям этого типа в модели холста существуют связи, фиксирующие относительное расположение холстов по оси Z. Их появление вызвано возможностью произвольного наложения холстов друг на друга. Пространство расположения холстов называют пространством двух с половиной измерений (twoand-half dimensions).

Холсту можно придать произвольную плоскую форму и менять ее в ходе выполнения программы; его можно передвигать в пределах холста-предка; для перекрывающихся холстов можно производить упорядочение по оси Z. Каждому холсту сопоставляется своя матрица преобразования координат, которую можно менять соответствующими операторами PostScript.

Построение изображения на холсте можно производить и до того, как будет вызвано его физическое воспроизведение на экране. Воспроизведение и удаление холста на экране могут выполняться многократно. Для того чтобы холст стал виден на экране, должен быть воспроизведен его предок.

Один из атрибутов холста определяет его прозрачность. Если холст прозрачен, то изображение, нарисованное на нем, оказывается на холсте — его предке, а то, что рисуется на самом холсте-предке (или любом другом нижележащем холсте), будет видно через прозрачный холст. Такие холсты используются для создания на нижележащих холстах зон чувствительности к воздействиям пользователя.

В NeWS PostScript для реализации холстов введен новый тип объектов (canvastype) и набор соответствующих операторов. Фактически каждый объект этого типа представляет собой словарь со специальным набором пар «имя—значение», являющихся атрибутами холста. Доступ к этим атрибутам осуществляется при помощи обычных средств PostScript, предназначенных для работы со словарями. Ограничения на доступ указаны в документации по системе NeWS. В графическое состояние интерпретатора языка введена переменная current canvas (текущий холст). Операторы манипулирования холстами выполняются или над текущим холстом, или над холстом, переданным явно через стек операндов.

Далее перечислены основные операторы манипулирования холстами (под холстами здесь понимаются соответствующие объекты языка):

newcanvas — создание нового холста;

createoverlay — создание нового холста специального типа (реализует метафору прозрачной целлофановой пленки);

readcanvas — считывает дискретное изображение в Sun-формате из файла и создает новый холст с этим изображением;

setcanvas — установка нового текущего холста;

currentcanvas — кладет текущий холст на стек операндов;

(eo) reshapecanvas — установка формы холста по текущей траектории;

(eo) clipcanvas — установка границ области отсечения изображений на текущем холсте;

clipcanvaspath — установка текущей траектории по границе текущего холста;

movecanvas — перемещение холста в плоскости XY относительно его предка;

(е) copyarea — копирует часть изображения на текущем холсте, заданную текущей траекторией;

damagepath — установка текущей траектории по границам нарушенных частей изображения на текущем холсте;

getcanvaslocation — возвращает расстояние холста до текущего холста в текущей системе координат;

canvastotop — операторы упорядочения canvastobttom по оси Z;

insertcanvasabove-оператор перекрывающихся insertcanvasbelow холстов.

В состав операторов раскраски языка PostScript были внесены изменения, относящиеся к работе с дискретными изображениями.

В NeWS операторы image и imagemask не встроенные, а реализованы на базе новых встроенных операторов: buildimage, imagecanvas и imagemaskcanvas. Аргументы и их интерпретация в операторе buildimage аналогичны оператору image, но результатом будет построение изображения на новом холсте, создаваемом в ходе выполнения оператора. По своему назначению операторы imagecanvas и imagemaskcanvas похожи соответственно на image и imagemask, но получают изображение не из PostScript-процедур, а путем его копирования с заданного холста на текущий холст.

Процессы

Как отмечалось выше, для обработки сообщений каждого клиента NeWS-сервер создает отдельный lightweight-процесс (далее просто процесс). Включением такого понятия в язык NeWS PostScript эта идея получила логичное завершение. Фактически тем самым PostScript был переведен в класс языков параллельного программирования. Однако целесообразность этого следует не только из наличия множества клиентов. В той или иной форме параллелизм потребовалось бы вводить для получения структурированных решений задач взаимодействия с пользователем.

При выполнении программы клиента может быть порождено целое дерево логически параллельных процессов. При этом каждый процесс протекает в своей среде, т.е. имеет свой графический контекст, стек вызовов, стек операндов и стек словарей.

Новый процесс создается вызовом оператора fork с процедурным объектом в стеке операндов. Значение этого процедурного объекта и будет программой нового процесса. Оператор fork формирует исходное состояние среды этого процесса, активирует его и возвращает процессу-родителю объект типа process.

Формирование среды нового процесса производится копированием текущего состояния среды процесса-родителя. При этом среда процесса-родителя частично становится разделяемой с процессом-потомком. Этот эффект возникает вследствие особенностей семантики операции копирования значений составных объектов (массивы, словари) в языке PostScript: копированию подвергается только один уровень вложенности, а значения элементов, являющихся составными объектами, становятся разделяемыми. Таким образом, у процессов появляется возможность взаимодействовать через общую память. Вместе с ней появляются и проблемы исключения одновременного доступа. Для их решения в NeWS PostScript введены средства для блокирования повторного входа процессов в критические секции программ.

К указанным средствам относится новый тип объектов, названных мониторами (monitor objects), и операторы для их создания (createmonitor), проверки текущего состояния (monitorlocked), вызова процедуры через монитор (monitor). Вызов любой процедуры через монитор будет блокировать прохождение через этот монитор других процессов.По завершению процедуры один из задержанных процессов будет разблокирован и сможет войти в свою критическую секцию.

В NeWS реализована non-preemptive-стратегия планирования процессов: однажды начавшись, процесс протекает до тех пор, пока не окажется блокированным. При этом в каждый момент времени активен только один процесс. Блокирование процесса происходит либо неявно (при операциях доступа к файлам), либо в результате выполнения некото-

рых операторов, перечисленных далее. Отметим, что выбор указанной стратегии планирования lightweight-процессов — это вопрос реализации, который может быть решен иначе в других версиях NeWS.

Подобно холстам, process-объекты реализованы как специального вида словари. Значения, ассоциированные с ключами такого словаря, определяют текущую среду и текущее состояние соответствующего процесса. Доступ к этим значениям осуществляется обычными средствами работы со словарями, но ограничен только чтением.

Для работы с process-объектами используются следующие операторы:

breakpoint задерживает текущий процесс;

continueprocess продолжает требуемый задержанный процесс;

currentprocess возвращает process — объект текущего процесса;

fork создает новый процесс-потомок текущего процесса;

killprocess уничтожает процесс;

pause задерживает текущий процесс на время, необходимое для того, чтобы дать шанс выполниться параллельно протекающим процессам;

suspendprocess задерживает требуемый процесс;

waitprocess ожидает завершения требуемого процесса и возвращает верхний элемент из его стека операндов.

При разработке графических интерфейсов возникают ситуации, когда функционирование не только интерфейса в целом, но и отдельных его элементов реализуется не одним процессом, а некоторым поддеревом процессов. Разработка таких элементов и управление ими упрощаются, если имеются встроенные средства манипулирования поддеревьями процессов. Для таких целей в NeWS PostScript введено понятие группы процессов (process group), реализуемое двумя операторами:

newprocessgroup создает новую группу процессов с текущим процессом как ее единственным членом, все его процессы-потомки при создании будут автоматически становиться членами этой группы;

killprocessgroup уничтожает требуемый процесс и всю группу процессов, к которой он принадлежит.

Интересы и события

Разделение контекстов и блокирующие операторы являются лишь частью тех средств, которые NeWS PostScript предлагает для реализации требуемых схем взаимодействия клиентов и отдельных PostScript-процессов. Теоретически и они обладают достаточно высокими моделирующими возможностями (можно реализовать, например, такой общий механизм, как семафоры). Однако главную роль среди базовых средств взаимодействия процессов в NeWS играют механизмы, реализующие модель событий.

Целесообразность введения этих механизмов можно объяснить следующими причинами. С одной стороны, в них реализовалось стремление авторов обеспечить модульность и гибкость прикладных систем. Другая причина заключается в том, что в терминах событий удобно представлять активность пользователя во взаимодействии с прикладной системой. Соответственно за счет выбора подходящей модели событий можно обеспечить единый базис для реализации как требуемых схем взаимодействий внутренних процессов NeWS, так и требуемых способов управления вводом от пользователя.

Характерными особенностями модели событий в NeWS являются введение понятия «интерес» процесса к событиям и возможность передачи через события любой информации. При этом интерес никак не связан с конкретными источниками генерации событий; допускается одновременное сосуществование множества интересов у одного процесса; имеются возможности динамического формирования, выражения и изменения интересов. Если несколько процессов выразили интерес к одинаковым событиям, то каждый из них получит возможность отреагировать на каждое такое событие.

Интересы и события представляются в NeWS PostScript структурированными объектами одного типа (event-объекты), реализованными как словари. Для генерации события надо создать новый event-объект, установить требуемые значения его полей и выполнить оператор передачи события в глобальную очередь механизма распределения событий (sendevent). Аналогично формируется

интерес: создается и настраивается event-объект, а затем выполняется оператор выражения интереса текущего процесса (expressinterest). В результате event-объект данного интереса будет помещен в один из списков интересов, которые анализируются механизмом распределения событий. Фактически каждый интерес является образцом событий, на которые процесс будет реагировать. Соответствие события интересу выясняется путем сравнения значения определенных полей их event-объектов по фиксированным правилам. Эти функции возложены на механизм распределения событий.

События отдельных выделенных типов генерируются самим NeWS-сервером в ответ на такие действия пользователя, как перемещение мыши, нажатие или отпускание одной из ее кнопок, нажатие клавиши, переход курсора через границу холста на экране и другие. При формировании этих событий сервер записывает в их event-объекты информацию о типе события, текущем времени, текущем положении курсора (холст и координаты). Заинтересованные процессы могут использовать эти данные в своих реакциях на события.

В NeWS отсутствует специальный тип таймерных событий, но при формировании любого события можно задать момент времени, когда оно должно произойти. Этой возможности достаточно для построения различных таймеров. Текущее время и время, когда произошло последнее событие, можно узнать, выполняя операторы currenttime и lasteventtime.

Для ожидания любого события из области своих интересов процесс должен выполнить оператор awaitevent. Этот оператор блокирует процесс до момента поступления соответствующего события, после чего копия event-объекта события будет положена на стек операндов данного процесса, а сам процесс будет разблокирован. Для избежания временных задержек перед выполнением awaitevent можно узнать число свершившихся событий из области интересов текущего процесса с помощью оператора countinputgueue. Анализируя поля полученного event-объекта, процесс выберет способ реагирования на него. Возможен другой подход к реализации реакции на событие — задание процедуры реагирования непосредственно при формировании интереса.

В отдельных случаях может потребоваться изменение или даже уничтожение ранее выраженного интереса. Изменение производится обычным способом, т.е. модификацией соответствующего event-объекта операторами доступа к словарям. Для уничтожения интереса предусмотрен специальный оператор (revoke-interest).

Таково общее представление о способе реализации модели событий в NeWS.

Построение программ клиентов NeWS

Для построения прикладных программ клиентов некоторого общего сервера требуется заранее определенный способ спецификации взаимодействий клиента с сервером, приемлемый в рамках выбранного языка программирования. Этот способ может быть сведен к использованию библиотек, макросредств либо к какой-то другой форме.

Например, построение программ клиентов на языке Си в системе X Window основывается на вызовах процедур библиотеки Xlib. Эту библиотеку называют Си-интерфейсом к X-протоколу связи клиентов и сервера. С точки зрения программиста, Xlib реализует такие базовые функции оконного сервиса X Window, как подключение к серверу, создание окон, графический вывод, работа с событиями. Фактически же большинство Xlib-подпрограмм транслирует вызовы в запросы уровня X-протокола и передает их по сети серверу. Он и выполняет семантические действия. Таким образом, в системе X Window спецификация взаимодействий с сервером основана на представлении базовых функций X-сервера в терминах Си-процедур и их аргументов. Для программиста изучение этого способа неотделимо от усвоения всей системы понятий о графических оконных интерфейсах, реализованной в X Window.

Аналогично X Window программы клиентов NeWS-сервера можно разрабатывать на различных языках программирования. NeWS также поддерживает определенный протокол связи клиентов с сервером и предлагает свою форму Си-

интерфейса к этому протоколу. Однако вследствие использования развитого языка для общения клиентов с сервером функциональная нагрузка и на протокол, и на интерфейс существенно уменьшилась по сравнению с X Window. Разработка NeWS-протокола фактически превратилась в выбор эффективного способа кодирования текстов PostScript-программ. Функции же интерфейса к NeWS-протоколу свелись к поддержке операций создания и разрыва связи с сервером, конструирования и передачи текстов PostScript-программ от клиента к серверу, ожидания и приема ответных сообщений сервера.

News-протокол

Реализация взаимодействий клиентов и NeWS-сервера в ОС UNIX основана на sockets-механизме (механизм гнезд). Такой выбор типичен в ситуациях, когда требуется реализовать взаимодействие процессов в UNIX-сетях. Общая схема использования sockets-механизма клиентами NeWS такова.

Для образования канала к серверу клиент должен создать новое гнездо и соединить его с гнездом сервера (адрес гнезда сервера хранится в переменной UNIX-среды). После установления соединения клиент может посылать сообщения серверу и принимать сообщения от него. Уничтожая свое гнездо, клиент разрывает связь с сервером. Для выполнения всех перечисленных действий вполне достаточно набора стандартных системных вызовов, соответствующих sockets-механизму.

На уровне NeWS-протокола сообщения, передаваемые от клиента к серверу и в обратном направлении, образуют байтовые потоки. Элементами потока, направляемого серверу, являются либо ASII-коды текста передаваемой PostScript-программы, либо коды компрессированных лексем этой программы. Эти две формы могут чередоваться в произвольном порядке в одном потоке. Аналогично строится и поток от сервера к клиенту. Однако содержание этого потока полностью определяется той PostScript-программой, которую клиент ранее передал серверу. Элементами содержания могут быть значения любых PostScript-объектов. При этом кодировка каждого значения определяется типом оператора вывода, использованного в программе (ASCII-коды — оператор print, компрессированная форма — taprint и typedprint).

Введение в News-протокол компрессированной формы передачи PostScript-текстов имеет своей целью только уменьшение временных затрат. В частных случаях достаточно использовать ASII-представление. Описание правил кодирования лексем занимает менее двух страниц в документации NeWS. В соответствии с этими правилами компрессированными могут быть лексемы, соответствующие целым и вещественным числам, именам встроенных операторов NeWS PostScript, а также некоторым именам, определяемым в программах. Для реализации этих возможностей NeWS-сервер поддерживает одну статическую таблицу для встроенных NeWS PostScript-объектов и ассоциирует с каждым клиентом по одной динамически изменяемой таблице для определяемых объектов. На основе порядковых номеров объектов в этих таблицах и строятся коды соответствующих лексем (один или два байта). Содержание таблицы клиента формируется в процессе выполнения его NeWS PostScript-программы. Для таких целей в NeWS PostScript был введен оператор setfileinputtoken.

Си-интерфейс NeWS

Взаимодействия клиента с NeWS-сервером можно специфицировать непосредственно в терминах системных вызовов и вызовов функций стандартной библиотеки ввода-вывода ОС UNIX. Однако для упрощения программирования на языке Си в состав программного обеспечения NeWS включены библиотека libcps и препроцессор CPS. Эти средства позволяют программисту оперировать абстракциями более высокого уровня, чем уровень NeWS-протокола.

С помощью библиотеки libcps программист освобождается от обязательного знания деталей организации канала связи клиентов с сервером. Отдельными процедурами этой библиотеки реализуются операции подключения клиента к серверу (ps_open_PostScript) и уничтожения соединения (ps_close

PostScript). Вызвав процедуру ps_flish_PostScript(), клиент может инициировать принудительную передачу буферизованной части байтового потока к серверу.

Препроцессор CPS определяет удобный способ синтаксического оформления операций конструирования и передачи сообщений от клиента к серверу. В общих чертах этот способ заключается в следующем.

Спецификации требуемых операций взаимодействия клиента с сервером выражаются в гибридной Си-PostScript-форме и называются интерфейсными процедурами. Заголовок такой процедуры строится по правилам языка Си, а ее телом является текст PostScript-сообщения. Если список формальных параметров в заголовке не пуст, то их имена могут быть включены в текст сообщения. При этом тип формального параметра должен соответствовать его использованию в PostScript-теле интерфейсной процедуры. В состав допустимых типов входят некоторые Си-типы (int, float, string), а также числа с фиксированной точкой (fixed), лексемы (token) и массивы символов (cstring). Вызов интерфейсной процедуры в Си-программе клиента приведет к подстановке параметров вызова в PostScript-тело этой процедуры и передаче его серверу. Посылаемое сообщение будет компрессированным.

Источниками сообщений, передаваемых клиенту сервером, могут быть любые lightweight-процессы, порожденные данным клиентом. Учитывая асинхронность этих процессов, программирование обработки выходного потока сервера на уровне NeWS-протокола не такая простая задача.

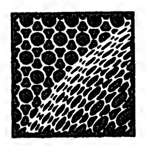
Препроцессор CPS позволяет абстрагироваться от представления выходного потока сервера в виде последовательности байтов или компрессированных лексем. Для этой цели препроцессор CPS определяет специальное синтаксическое понятие «интерфейсная функция». В виде интерфейсных функций задаются спецификации операций типа «послать сообщение — принять ответ». При этом результаты работы сервера представляются как очередь тэгированных пакетов. Обработка же сообщений сервера основывается на операции проверки соответствия первого пакета очереди требуемому тэгу и операции задержки на время, пока очередь пуста.

Каждый тэгированный пакет имеет структуру последовательности компрессированных лексем, первый элемент которой является тэгом. В качестве тэгов используются целые неотрицательные числа. На стороне сервера тэгированные пакеты формируются lightweight-процессами данного клиента с помощью операторов вывода tagprint и typedprint.

Синтаксически каждая интерфейсная функция состоит из двух частей — интерфейсной процедуры и описания тэгированного пакета. Первая определяет передаваемое сообщение, а вторая — принимаемое. В таких спецификациях параметрами интерфейсной процедуры могут быть и указатели. Из списка имен этих параметров-указателей и значения тэга строится описание пакета.

Вызов интерфейсной функции в Си-программе клиента имеет следующую семантику. На первом этапе выполняются те же действия, что и при вызове интерфейсных процедур. После же посылки сообщения серверу производится анализ очереди пакетов от сервера. Если очередь пуста, то текущий процесс блокируется до поступления любого сообщения от сервера. Если тэг первого пакета очереди не равен тэгу данной интерфейсной функции, то результатом вызова будет логический нуль. Иначе выполняется распаковка пакета и запись его элементов по указателям, заданным в вызове. Возвращаемым значением в этом случае будет логическая единица.

Путем разработки набора интерфейсных процедур и функций программист может создать свой Си-интерфейс у каждого клиента NeWS-сервера. В соответствии с требованиями препроцессора CPS все эти определения должны помещаться в отдельный файл, называемый файлом спецификации интерфейса. В обработке таких файлов и заключается функция препроцессора. В ходе обработки препроцессор CPS транслирует интерфейсные функции и процедуры в форму вызовов подпрограмм библиотеки libcps. В результате создается h-файл, который включается в текст Си-программы клиента директивой include.



В.В.БОРИН

Путешествие в страну «Информатика»

Давайте познакомимся: меня зовут Коля. Я учусь в 5-м классе. Школа наша обычная, средняя, «без всяких уклонов», как говорит папа. Он имеет в виду, что школы могут быть с более углубленным изучением математики, физики или даже иностранного языка. Но тем не менее у нас уже есть свой компьютер — маленькая, как домашний телевизор, ЭВМ. Мерцает огоньками и выдает текст на экране. Точь-в-точь как на большом футбольном табло в Лужниках. Только там буквы набирает незнакомый дядя, который сидит где-то в кабинке под трибунами, а тут ты сам. Включаешь тумблеры, отжимаешь клавиши и... поехали! Но этого мало: ты не просто набираешь текст, ты — командуешь машиной. И она тебя слушается, понимает. Можно ли в 5-м классе научиться работать на ЭВМ? Папа говорит: не только можно, но и нужно! Иногда он шутит: «Ты уже старичок. Надо было еще раньше начинать...»

Я смотрел по Центральному телевидению выступление академика Андрея Петровича Ершова. Он рассказал удивительную вещь. Оказывается, дети в 4—5 раз быстрее, чем взрослые, осваивают работу на ЭВМ. У них там в Сибири уже давно школьники ходят в ВЦ (так сокращенно называется вычислительный центр) при Академии наук, как в школьный класс. Правда, говорят, ходят не все, а «особо подающие надежды». По-моему, каждый из нас подает какие-то надежды, к чему-то стремится. Один мечтает стать инженером, другой — врачом, третий — композитором, четвертый — писателем. К тому времени, как мы подрастем, на каждом рабочем месте, даже на письменном столе писателя будет стоять маленькая черная коробочка — микро-ЭВМ. Говорят, Лев Толстой переписывал свое знаменитое творение «Война и мир» пять раз. Если бы в то время были компьютеры, его правка была бы намного проще и не требовала от гения массы черновой работы.

В нашей серии будет рассказано, как я сам встретился с компьютером, составил первую в своей жизни маленькую программку. Мы совершим вместе путешествие в страну «Информатика», узнаем массу интересных вещей из области применения компьютеров сейчас и в самом недалеком будущем.

Вторая часть выпусков серии называется «Для тех, кто отважился идти дальше». Тут уже мы познакомимся с различными языками программирования, на которых «говорят» современные компьютеры. Узнаем про ЭВМ 5-го поколения и даже про системы с искусственным интеллектом. Пусть вас не пугает это мудреное название. По себе знаю: и школьнику вполне под силу разобраться, что такое интеллектуальный компьютер. А они уже на подходе.

Итак, ребята, в путь! Как в песне поется: «Дорогу осилит идущий...»

ВВЕДЕНИЕ

Сколько у тебя друзей? Давай считать: Саша, Гена, Володя, Артур, Лена... Весь класс? Нет, настоящими друзьями могут стать не все. И ты это уже, наверное, понял. Есть такие друзья, за которых ты готов и в огонь, и в воду. Есть люди, на которых ты хочешь походить, подражать их смелости и благородству. Их ты мог встре-

тить и в книгах, и в кинофильмах, и в театре юного зрителя. Например, Тимур — герой замечательной повести Аркадия Петровича Гайдара. И видимо, не случайно изобретатели новой школьной ЭВМ назвали ее «Тимур». Недавно у «Тимура» появился собрат — тоже микрокомпьютер. Изготовили его в другом министерстве и, видимо, поэтому дали ему имя не «Коля Колокольчиков» или «Женя», а... «Агат». Агат, если ты помнишь, это минерал, обычно используемый для декоративных целей. К компьютерам типа «Тимур», «Агат», «Микроша», кроме приставки «микро-» (по-гречески она означает «малюсенький»), иногда добавляют слово «персональный». Что бы это значило?

В последние годы за пультом ЭВМ все чаще можно встретить таких людей, как врачи, лингвисты, режиссеры и даже музыканты, которые раньше никогда не занимались программированием. Да и ты знаешь, что у нас в стране идет сейчас компьютерный всеобуч. В законе о реформе средней общеобразовательной школы прямо сказано: «Вооружить учащихся знаниями и навыками использования современной вычислительной техники, обеспечить широкое применение компьютеров в учебном процессе». Но можно ли «вооружить» школьника такими знаниями и навыками? Ведь до самого недавнего времени, чтобы освоить ЭВМ, надо было пять лет проучиться в институте да потом еще не один год набираться опыта в программировании. И вот тут на помощь приходят персональные компьютеры. В отличие от своих прежних собратьев они не требуют от человека специальных «компьютерных» знаний. Достаточно владеть элементарной компьютерной грамотой. Для начала от тебя требуется только нажимать на клавиши и читать текст на экране. Академик Евгений Павлович Велихов говорит, что в недалеком будущем человеку достаточно будет нажать кнопку с надписью HELP («ПОМОГИ!»). Всему остальному тебя научит компьютер. Научит, как говорят специалисты, «в режиме диалога», т.е. в форме разговора, беседы, которую ведут два близких человека. Компьютер превращается в настоящего друга, с которым можно вести доверительную беседу, не стесняясь признаться, если что не понял.

Сейчас насчитывается более трех тысяч областей применения персонального компьютера. С одной стороны, ему доступно делать то, что могут его старшие собратья (например, решать математические задачи с фантастической скоростью). С другой стороны, ежедневно приходят сообщения о все новых и новых профессиях персональной ЭВМ. Например, компьютер — записная книжка. Или компьютер — рабочая тетрадь. Он с успехом может заменить черновик для решения математических задач (ведь иногда очень важно не только получить окончательный результат, но и сохранить все промежуточные вычисления), журнал погоды по географии или природоведению. В недалеком будущем, связавшись с помощью своего персонального компьютера с базой данных городского метеоцентра, ты сам сможешь предсказать погоду на завтра. Персональный компьютер поможет тебе лучше узнать родной язык (ты сам сможешь легко «загнать» в его память список наиболее трудных для правописания слов или исключений). То же самое и при изучении иностранного языка. Уже сейчас есть в продаже маленькие, с губную гармошку, пластмассовые приставки — синтезаторы речи. Компьютер с помощью синтезатора произносит новое, например, английское слово. Школьник, услышав его, должен набрать его по буквам на клавиатуре. Если он ошибается, компьютер начинает пищать электронным голосом: «Пробуй снова! Пробуй снова! Ты не знаешь это слово». В газетах уже сообщалось, что известны случаи, когда трехлетние малыши самостоятельно осваивали русскую грамоту, играя с персональной ЭВМ. Школьник же сможет с помощью персонального компьютера провести эксперименты по физике и химии, оформить домашнее задание, в том числе написать сочинение без единой ошибки и помарки. В отсутствии хозяев персональный компьютер сможет отвечать на телефонные звонки, поддерживать в квартире заданную температуру и влажность, поливать

цветы, кормить в строго определенное время рыбок в аквариуме. А к приходу хозяев он приготовит обед, постирает и выгладит белье. Персональная ЭВМ, подключенная не только к телефону, но и к телевизору, электроплите, швейной или стиральной машине, а то и к простому утюгу или молотку, приводит к удивительным превращениям. Обычная вещь приобретает совершенно новые качества. Такие встроенные ЭВМ берут на себя различные обязанности домашних роботов, давно уже кочующих по страницам фантастических рассказов. Через персональную ЭВМ можно в мгновение ока получить подборку самых свежих новостей по любой заданной теме, заказать билет на поезд или самолет, просмотреть на экране текст и рисунки из любой редкой книги (даже если она имеется только в Государственной библиотеке имени В.И.Ленина в Москве). А для самых маленьких членов семьи компьютер почитает вслух любимую сказку и даже покажет заказанную серию из «Ну, погоди!». Хочешь сыграть в шахматы, баскетбол, хоккей, пинг-понг и даже в большой теннис — пожалуйста, компьютер к твоим услугам. Так разве нельзя после всего этого назвать персональный компьютер настоящим членом семьи?!

Это про то, что может компьютер дома, а как изменит компьютерный всеобуч вид каждого класса и саму учебу?

За каждой партой (да, именно партой, ученые говорят, что она лучше подходит для учебы, чем стол) появится твой новый друг «Тимур» или «Агат». На учительском столе будет стоять такой же дисплей с клавиатурой, но к нему добавится небольшая электронная приставка-пульт управления всеми классными дисплеями. С его помощью учитель сможет в полной тишине (ведь каждый занят своей задачей) задавать вопросы, давать комментарии, ставить оценки — иными словами — управлять процессом обучения. Например, идет контрольная по русскому языку. На экране-дисплее, установленном на каждой парте, высвечивается слегка «деформированный» текст. Что это значит? Иными словами, текст выдается с пропуском трудных для правописания букв или знаков препинания. Задача ученика — в соответствии с изучаемым правилом вставить в экранный текст пропущенные буквы и знаки. Например, вот такое предложение:

УЛ ТАЮТ ЖУРАВЛИ И НИ КИЕ ОСЕННИЕ ОБЛАКА ЗАВ ЛАКИВАЮТ НЕБО. Поскольку ученик избавляется от нудной работы, связанной с переписыванием всего текста, изучение правописания можно вести более целенаправленно и в быстром темпе.

Более сложный вариант: редактирование сочинения или изложения и его копирование с помощью компьютерного печатающего устройства («принтера»).

Если ты ответил на все вопросы компьютера правильно, на экране вспыхивает надпись:

«ТЫ МОЛОДЕЦ! ЗАДАНИЕ ВЫПОЛНЕНО ВЕРНО. ОЦЕНКА 5».

Однако следует сразу же сказать: наш обычный, разговорный язык недоступен персональной ЭВМ. Любой вопрос, любую команду ей надо давать не на человеческом, а на так называемом алгоритмическом языке. Как это делать, учит новая, бурно развивающаяся наука — ИНФОРМАТИКА.

ИНФОРМАТИКА ВОКРУГ НАС

Информатика — это наука, изучающая законы, по которым происходит получение, обработка, накопление и распределение информации. Как можно себе представить, что такое информатика и информация? Если сравнить информацию с залежами полезных ископаемых, то информатика — это технология получения из «сырой» руды конечного продукта — информации.

Вспомним технологию получения металла. Его сначала нужно добыть из недр Земли, очистить от ненужных примесей, загрузить в вагранки. Затем в строгом соответствии с предписаниями технологического процесса получают выплавленный металл, «отливки». Они после очистки поступают последовательно в различные производственные цехи: кузнечно-прессовый, ковки, штамповки — для механи-

ческой, термической или даже гальванической (т.е. электрической) обработки. И только в завершение этого гигантского производственного конвейера и выходит, как говорят экономисты, конечный продукт — то ли легкая, но очень прочная обшивка сверхзвукового самолета, то ли кровельное железо для крыш. Так и информатика занимается сложными процессами обработки — только не земной, а словесной руды — массы всевозможных, зачастую не очень нужных сведений, перемешанных с цифровыми данными, и «выплавляет» из них очищенную от примесей («шумов») необходимую информацию.

Следует подчеркнуть: несмотря на определенную схожесть информации с продуктами окружающего нас материального мира (ведь фюзеляж самолета — веществен, материален), она имеет и важное отличие. Информация — нематериальна, незрима, ее не пощупаешь, она существует только в нашем сознании. Другое дело, что носители информации — листы тетради, страницы книги, магнитофонная лента и даже запотевшее стекло в электричке, на котором кто-то вывел свои инициалы, — материальны. Но сама информация бестелесна, она существует только в нашей голове, идеально. Однако такая бестелесность еще совсем не говорит о том, что информация — это мираж, воображение, не стоящее ломаного гроша. Вспомним увлекательную книжку английского писателя Р.Стивенсона «Остров сокровищ». «К черту деньги! — закричал слепой. — Я говорю о бумагах Флинта...»

Там, на затерянном в океане крошечном острове, пираты искали в таинственном сундучке не пригоршни золотых монет («Пиастры! Пиастры! Пиастры!»), а нечто гораздо более ценное — информацию. Простой и невзрачный с виду клочок пергамента с дюжиной нацарапанных значков превращал его владельца в сказочного богача и, как думали пираты, в самого счастливого человека на свете.

Сейчас информацию научились считать, как выплавленный чугун, выращенный хлеб. Установлен даже своего рода «грамм информации» — 1 бит. И это неслучайно. Без наведения учета, строгого порядка в окружающем нас океане информации современному человеку не обойтись. Изобретение книгопечатания, а затем телеграфа, телефона, радио и телевидения привело к тому, что на каждого из нас ежедневно обрушиваются настоящие лавины информации. Недаром говорят об «информационном взрыве» конца XX века. Но возможности любого человека переработать, «переварить» столь обильные «порции» информации ограниченны. И вот тут-то ему на помощь приходит маленькая электронная машинка — персональный компьютер. Компьютер переваривает информацию с фантастической скоростью, может работать без устали днем и ночью, облегчая жизнь своему владельцу, помогая решать сложные задачи. Но чтобы дать задание компьютеру, человек должен обладать еще одной грамотой, кроме той, которую вы изучали в первом классе. Эта вторая грамота называется компьютерной. Так же, как цивилизованному человеку еще совсем недавно достаточно было уметь читать и писать, так и каждому жителю нашей планеты на пороге 2000 года жизненно необходимо знать и компьютерную грамоту. Недаром родилось крылатое выражение: «Компьютерная грамотность — пропуск в XXI век!»

Но уже сейчас есть такие вещи, такие области человеческой деятельности, в которых без ЭВМ никак не обойтись. Вот, скажем, прогноз погоды. Он нужен морякам, геологам, летчикам, тем, кто выращивает урожай. Предупредить о грозе, снежной лавине, селевом потоке, о ранних заморозках или развитии градового облака, а то и наводнении — это сберечь миллионы рублей, а подчас отвести злую беду от человеческой жизни, причем не от одной. Да и школьнику тоже небезразлично, какая будет завтра погода. Сделать это просто — достаточно включить дома телевизор. В сжатом, спрессованном виде прогноз погоды регулярно сообщает информационная программа «Новости». Но чтобы высветить на наш домашний дисплей и эти лаконичные сведения, компьютеру, а точнее огромному вы-

числительному центру с его десятком мощных суперкомпьютеров, пришлось немало потрудиться. Ведь требовалось не просто собрать всю имевшуюся метеорологическую информацию (исходные данные) со спутников и тысяч метеостанций, раскиданных по всей планете, перекачать ее в чрево компьютеров. Требовалось проанализировать эти исходные данные на основе сложных математических уравнений, описывающих взаимодействие еще более сложных физических процессов, которые происходят в атмосфере и океане. Проделать этот огромный объем работы, перелопачивая Гималаи информации вручную, с помощью арифмометра, не говоря уже об авторучке и счетах, просто невозможно. Даже если усадить за расчеты все население Земли, прогноз погоды на завтра был бы готов только через... месяц. А ведь прогнозные данные надо еще нанести на географические карты, привязать к определенным ориентирам, времени суток, движению циклонов и антициклонов, грозовых фронтов. Но и этого мало: надо еще своевременно (то есть заблаговременно) доставить эти карты с нанесенной прогнозной метеообстановкой многочисленным заинтересованным лицам — пользователям. Теперь тебе ясно, что без помощи компьютера тут не обойтись. Не обойтись как при вычислениях, так и при тиражировании великого множества карт. Современный компьютер снабжается дополнительной приставкой — графопостроителем. Он позволяет автоматически печатать на огромных листах бумаги, причем всеми цветами радуги, все то, что отображает компьютерный дисплей. В том числе изобары, изотермы и другие предвестники грозных погодных явлений.

Или ты летишь на каникулы, скажем, в Крым. Время каникул и летних отпусков у взрослых — самая горячая пора для билетных кассиров. И тут они спасовали без помощи ЭВМ. Пришлось создать даже разветвленные компьютерные системы «Сирена-2» и «Экспресс-2» в масштабах всей страны. Одна из них обслуживает Аэрофлот, другая — железные дороги. Мощный компьютер «Сирены-2» хранит в своей магнитной памяти номера простых и специальных авиарейсов, состав многочисленных летающих экипажей. Последнее для авиации, точнее, для безопасности полета имеет немалое значение. Ведь далеко не каждый пилот обладает высшей квалификацией, некоторые еще «не доросли» до полетов в сложных погодных условиях. Таким образом, экипажи даже однотипных самолетов далеко не взаимозаменяемы. В память компьютера вводятся также сведения о свободных и бронированных местах во всех аэропортах страны. Но и этого мало. Информация в «Сирене» не стоит на месте, она «дышит», меняется при каждой покупке билета, при назначении дополнительных рейсов и, разумеется, при задержке вылета из-за неблагоприятных метеоусловий. Иными словами, в случае даже не плохой, а просто неважной погоды подниматься в воздух некоторым самолетам категорически запрещается: не позволит летная подготовленность экипажа. Об этом обязательно должен знать компьютер. Но и это еще не все: любая информация должна по первому требованию билетного кассира выдаваться на экран его рабочего пульта или, как говорят, терминала. Наконец, бывают случаи, когда пассажир выбирает авиарейс или маршрут с пересадками. Раньше это сильно осложняло жизнь и пассажиру, и целой армаде билетных кассиров в аэропортах пересадки. Теперь же печатание даже такого нестандартного авиабилета происходит автоматически. Компьютер сам проставляет в соответствующих местах не только дату, но и часы и минуты взлета и посадки на много дней вперед. Именно так работает многокомпьютерная система продажи авиабилетов под названием «Сирена-2».

Недавно газеты сообщили, что группа советских геологов была удостоена государственной премии. Наши ученые разработали новую методику поиска залежей черного золота — нефти. Соавтором новой методики может быть назван и компьютер.

Нефть — это хлеб нашей индустрии. Она нужна и реактивным лайнерам, и вертолетам, и мощным БелАЗам, и маленьким юрким «москвичам». Она —

основа основ всей химической промышленности, производящей широкий спектр так необходимых изделий — от лавсана и искусственного шелка до сложнейших лекарственных препаратов. С каждым годом находить нефть становится все труднее и труднее, а ее добыча обходится все дороже — основные нефтяные месторождения «прописаны» в малообжитых северных районах. Природа не спешит расстаться со своим богатством. Можно продырявить пробными скважинами территорию, равную нескольким Франциям, и все впустую — нефти будет мало или она окажется низкого качества. Мечта геологов, чтобы земля была прозрачна, как стеклышко. И вот ученые-геологи обратились к компьютеру. Именно с его помощью земной шар стал чуточку «прозрачнее». «Прозрачнее» — благодаря умело составленным машинным программам. Компьютер-геолог решает не простую, а вероятностную задачу: он указывает на своем дисплее те точки, где шансы на успех нефтяников-буровиков предпочтительнее. Не нужно говорить, что это экономит миллиарды рублей и едва ли не самый важный ресурс человечества — время.

Недавно в повседневный обиход пришло новое племя компьютеров — так называемые микропроцессоры. Микропроцессор — сверхминиатюрная ЭВМ на одном кристалле, умещающаяся в обычном наперстке. Такой малюсенький компьютер можно встроить в стиральную машину, токарный станок и даже обычный молоток. И тут происходит настоящее чудо: стиральная машина, станок и молоток приобретают элементы человеческого интеллекта, они становятся «умнее» своих беспроцессорных собратьев. К примеру, микропроцессорный молоток автоматически регулирует силу удара, причем на торце вспыхивают (как в наручных часах, работающих на жидких кристаллах) цифры, которые указывают требуемое, то есть наилучшее (или, как говорят, оптимальное) в данных условиях давление, а значит, и оптимальную силу удара.

Но настоящее компьютерное царство — на борту космического корабля. Здесь используется не одна, а сразу несколько ЭВМ, завязанных в единый комплекс. Они с недостижимой для наземных условий точностью управляют ориентацией орбитальной станции. Ведь «крылышки» станции — батареи солнечных элементов — должны перехватывать максимум световой энергии, чтобы космонавты не испытывали недостатка в электрическом токе. Электрический ток тем и хорош, как ты уже знаешь из физики, что его легко преобразовать в любой вид энергии. А она требуется на станции для разных целей: для обогрева, освещения, приведения в действие механических агрегатов, космической сварки, химических и многих других экспериментов, проводимых в условиях невесомости. Невесомость — уникальное явление, его никак не удается хотя бы в течение получаса воспроизвести в земных условиях. В космосе невесомость позволяет получить идеально чистый слиток металла или уникальный лекарственный препарат. Но есть такие земные вещи, без которых даже самым отважным землянам на орбите не обойтись. Это кусочек земной атмосферы, воздух, которым дышат космонавты. Его газовый состав и чистота бдительно контролируют бортовые ЭВМ.

На борту космических кораблей появилось новое поколение компьютеров, в которых нет терминала с его привычной клавиатурой, напоминающей пишущую машинку. Но как тогда связаться с ЭВМ, отдать нужный приказ? И вот теперь космонавт отдает ей команды не вручную, набирая определенный кусочек текста, а... голосом. Такие интеллектуальные компьютеры уже понимают человеческую речь. Правда, понимают пока не все слова и часто «просят» повторить: ведь дикция, манера произношения у каждого человека своя.

Уже прошел испытание компьютер, который позволяет экипажу управлять полетом космического корабля движением своих... зрачков. Кажется, зачем это нужно, ведь есть же на пульте отлично вписывающаяся в ладонь ручка управления, наконец, многочисленные кнопки, клавиши и тумблера? Но не следует забывать, что каждый космический полет — это езда в не-

знаемое. Далеко от Земли экипаж могут подстерегать и так называемые нештатные, а попросту говоря, опасные ситуации. В такой ситуации бывает непросто дотянуться рукой до нужного переключателя или отжать клавишу из-за внезапно возникших перегрузок. Другое дело приказать глазами. Оказывается, компьютер способен и на такое — материализовать этот нематериальный приказ.

Иной читатель может спросить: все это хорошо, может быть, даже где-то интересно. Но пригодятся все эти компьютеризованные предметы — машины, станки, орбитальные станции, наконец, — тем, кто решил связать свою жизнь с техникой, стать инженером. Да, им разбираться в компьютерной электронике необходимо. Но я не собираюсь стать инженером. Мне нравятся стихи, музыка. Зачем мне знать, как работает компьютер, тем более учиться писать машинные программы?

Правы ли такие ребята? Мне кажется, нет, и вот почему.

Сейчас с каждым годом становится все яснее: в наш век даже в искусстве нельзя обойтись без того, что дает нам НТР — научно-техническая революция. Кстати, это не мои слова, не автора — в определенной степени заинтересованного лица. Почти слово в слово то же сказал — кто бы вы думали? — выпускник хорошо известного в музыкальном мире Государственного музыкально-педагогического института имени Гнесиных.

...В актовом зале института идет дипломный концерт. Выпускник представил на суд экзаменационной комиссии театрализованную ораторию. Она написана по мотивам знаменитой горьковской «Песни о Буревестнике». Вот хор, сопровождаемый симфоническим оркестром, величаво пропел: «То кричит пророк победы. Пусть сильнее грянет буря!» И тут по всему пространству концертного зала вдруг заметались тревожные и зовущие звуки. Они обрушивались на слушателей то с одной, то с другой стороны. С нарастанием падали сверху, чтобы мгновенно раствориться и потом вновь возникнуть откуда-то снизу. Казалось, стены зала раздвинулись, чтобы дать простор беспокойной невидимой птице — символу грядущей Революции.

«Что? Что за инструменты звучат?» — недоумевали в экзаменационной комиссии. И лишь немногие знали, что столь необычный эффект создан с помощью компьютерной музыки. Ее автор — вчерашний школьник из города Снежное Донецкой области Виктор Ульянич.

Еще на третьем курсе он стал лауреатом Всесоюзного конкурса молодых композиторов. И не случайно: как раз в то время он занялся программированием компьютерных музыкальных систем. Сейчас Виктор первый в истории нашей страны аспирант-композитор Вычислительного центра Академии наук. Тема его диссертации — «Применение компьютеров в музыкальном творчестве». По правилам, у каждого диссертанта обязательно должен быть научный руководитель. Так вот у Виктора их два. Один — кандидат искусствоведения доцент гнесинского института Ю.Н.Рагс, известный своими исследованиями творчества Скрябина. А другой научный руководитель — кандидат физико-математических наук А.С.Тангян, ведущий семинар «Кибернетические проблемы музыки», организованный в Вычислительном центре Академии наук. Так будущий ученый «синтетического профиля» Виктор Ульянич претворяет в жизнь дерзкую мечту композиторов нашего века — КОМПЬЮТЕРОМ ГАРМОНИЮ ПОВЕРИТЬ. А ведь эта идея вдохновляла и нашего великого поэта Александра Сергеевича Пушкина. Только он устами своего героя хотел поверять гармонию алгеброй. Просто не было в ту пору у его современников такого надежного, все понимающего друга и собеседника, собранного на микросхемах, — персонального компьютера.

Е 84 **Если бы** у Марианны была ЭВМ. — М.: Знание, 1993. — 48 с. — (Новое в жизни, науке, технике. Сер. «Вычислительная техника и ее применение»; № 1).
ISBN 5-07-002613-5

Предлагаем читателям несколько новых рубрик и среди них — «Искусственный интеллект». А для юных пользователей мы совершим путешествие в страну «Информатика», где ребята откроют для себя много нового и интересного.
Материал рассчитан на широкий круг читателей.

ББК 32.81 **2404000000**

Научно-популярное издание

ЕСЛИ БЫ У МАРИАННЫ БЫЛА ЭВМ...

Редактор *И.В.Кащенков*
Мл.редактор *Н.А.Сергеева*
Художник *В.Н.Конюхов*
Худож. редактор *И.А.Емельянова*
Техн. редактор *Т.В.Луговская*
Корректор *Е.В.Юхимец*

ИБ № 12255

Подписано к печати 25.02.93. Формат бумаги 70×100 1/16. Бумага офсетная. Усл. печ. л. 3,90. Усл. кр.-отт. 8,45. Уч.-изд. 3,85. Тираж 14814 экз. Заказ 299. С—8. Издательство «Знание». 101835, ГСП, Москва, Центр, проезд Серова, д. 4. Индекс заказа 934701. Отпечатано с оригинал-макета издательства «Знание» на Ордена Трудового Красного Знамени Чеховском полиграфическом комбинате Министерства печати и информации Российской Федерации.
142300, г. Чехов, Московской области

Индекс 70195

Адрес подписчика:



Наш адрес:
101835,
Москва,
Центр,
проезд
Серова, 4